THRUSTMASTER



HOTAS COUGAR

MANUEL DE REFERENCE

Beta 1 Traduit par



INTRODUCTION

Remerciements!

Tout d'abord, nous tenons à vous FELICITER et à vous REMERCIER pour votre aquisition de ce superbe HOTAS Thrusmaster Cougar ! Ce controlleur le plus puissant, et d'une extrême précision, que vous avez désormais entre vos mains est le fruit de 2 années d'etudes minutieuses et de développement, menées dans l'unique but de créer un contrôlleur précis pour les jeux de simulation qui corresponderait aux demandes les plus exigeantes des joueurs. Nous l'avons fait , digne héritier de la série des FLCS et des F-22 PRO. Toutefois, ce challenge à créer le digne successeur des séries de controlleurs précédents fut un plaisir pour nous tous, faire naitre avec succès cette divine chose s'avéra être beaucoup plus qu'un simple défi - ce n'etait pas une mince affaire !

Quand ce projet fou débuta, deux choix se présentaient à Thrusmaster : soit effectivement mettre à jour les manettes existantes en reprenant et en améliorant leurs caractéristiques ou alors partir de zéro et inventer entièrement un nouveau produit. Evidemment en adoptant la seconde solution, Thrumaster s'est engagé dans la voie la plus difficile. Pour quelle obscure raison ? Tout simplement parce que toutes les fois ou Thrusmaster a décidé de réaliser de nouveaux controllers HOTAS, ces nouveaux joysticks ont atteint à chaque fois un nouveau degré de qualité hissant les limites loin devant les productions du moment, pour ces raisons il etait temps de montrer avec le HOTAS Cougar ce qu' un véritable joystick hardcore se doit-être.

Le résultat de ces interminables séances de reflexions et de nuits-blanches , c'est ce controlleur, extremement réaliste, incroyablement versatile, etonnamment facile d'emploi et monumentallement puissant. Sans aucun doute, avec cette collection acérée de fonctionnalités inégalées à ce jour, comme les manches interchangeables, la possibilité de programmation avancée (exceptionnelement facile), la simplicité absolue d'une connexion plug'n'play, ajouté au nombre impressionnant (et au poids) des diffèrentes parties assemblées, font de ce superbe Hotas Cougar le meilleur choix pour les années à venir.
Cette réussite est aussi un succès pour la communauté tout entière des adeptes de la simulation, croyez nous, cette divine chose n'est ni une chose bon marché, ni imaginaire, ni un truc destiné aux bornes d arcade mais plutôt la réplique hardcore des commandes d'un F-16 concus pour ceux qui ne demandent rien d'autre que ce qui se fait de mieux. C'est réellement l'ultime controlleur haut de gamme qui laissera tout le monde du meme avis : un must!!
Bien entendu, cette chose n'aurait pas vu le jour sans les soutiens et les encouragements des personnes que nous avons rencontrés aucours des expositions, avec qui nous avons discuté dans les forums, échangé longuement

de très nombreux mail, alors merci à vous tous de nous avoir tant soutenu ! N'oublions pas non plus toutes les personnes à qui nous devons tout spécialement nos remerciements; tout d'abord toute l'équipe de Thrusmaster/Guillemot qui ont travaillé sur ce projet, ainsi que les beta testeurs, qui ont fait un superbe travail à rechercher tous les bugs, et experimenter un équipement au péril de leur vie…Enfin merci à nos amis et à nos familles qui ont eu de la patience avec nous tout au long de ce projet.

Allez, assez de palabres, revenons au sujet principal !

Si vous avez déjà utilisé l'ancienne gamme des commandes Thrusmaster pour simulateurs de vol, la puissance du COUGAR va vous intimider En fait, certaines caractéristiques *semblent* ne pas avoir changer, ne vous trompez pas, c'est completement nouveau et comme vous êtes un utilisateur averti de simulateur de vol, mieux que quiconque, vous serez à même d'évaluer toute la substance du travail difficile que nous avons mis à l'intérieur afin de vous proposer une puissance sans précédent.

Avec ce nouveau produit, nous avons perfectionné et innové sur tous les points, mécaniquement, electroniquement et au niveau des composants logiciel.
Attendu que proposer un controller plus convivial que ceux qui ont déjà 7 ans n'est pas fait pour prouver que c'est extremement difficile, ce que nous avons voulu faire, c'est donner à tout joueur même débutant, la possibilité d'utiliser à son maximum toutes les capacités brutes de ce controlleur. En fait, la sortie de Microsoft Windows a permis l'utilisation de controllers compatibles HID, des manettes plus'n'play qui ne demandent pas une configuration particulière pour fonctionner. Et bien c'est exactement que le HOTAS Cougar vous propose, branchez le et jouez à votre jeu favori…n'est-ce pas simple.

Et bien, ceux sont des bonnes nouvelles, mais beaucoup parmis *vous* qui ont acheté le HOTAS Cougar pour les possibilitées de programmation avancée, lui demandent plus qu'être seulement un *superbe* controller.Soyez donc attentifs, nous y arrivons! Pour garantir la précision inégalée qu'il propose, le HOTAS Cougar offre les mêmes caractéristiques que le F-22… et même plus, encore plus de choses que vous pourrez apprendre tout au long de la lecture de ce manuel de référence. Les caractéristiques de programmation que nous allons aborder ci-après sont en fait si puissantes et si compréhensibles qu'ils vous permettront d'optimiser tous vos programmes pour tirer partie de vos jeux, ou même de corriger des bugs ou des fonctions manquantes dans vos simulateurs. La meilleure facon des les découvrir et de les assimiler (ceci-dit sans vous provoquer une migraine et des cauchemars dus à la programmation) est simplement de lire calmement et de facon attentive les étapes importantes du manuel et de vous donner le temps de vous poser la question "Bon, que puis-je faire avec *cela* ?"

Comme souligné précedemment, ce document est-ce que nous appelons le “Manuel de Référence. N’importe quelle personne qui avait l’habitude des manuels pour les controlleurs précèdents F-16 FLCS, TQS et F-22 PRO va apprécier les diffèrences qui les séparent du présent manuel de référence. Il contient tout ce que vous avez besoin de savoir pour régler votre controlleur, en utilisant le panneau de contrôle du Cougar, pour apprendre les bases de la programmation du Cougar à travers des informations détaillées sur la syntaxe du language de programmation qui mettent le HOTAS Cougar à part des autres controlleurs programmables. En plus de ce manuel, des aides très détaillées, des assistants, des tutoriaux et de nombreux autres outils utiles et intuitifs sont fournis à l’intérieur du logiciel de programmation. En fin de compte, nous pensons avoir réussi à proposer la documentation la plus compréhensible jamais livrée avec tout autre controlleur

Ne vous faites pas d’illusion à propos de ce HOTAS Cougar et du manuel de référence, c’est un controlleur hardcore dans les régles de l’art, et ce manuel de référence est composé de sections et d’instructions que vous voudrez lire plus d’une fois. Mais l’une des raison pour lesquelles le manuel est des plus détaillés est que nous avons été vivement informés que le précédent manuel etait trop incomplet et en conséquence le controlleur trop difficile à comprendre. Vous allez donc trouver que ce manuel est très facile à lire, et présente les choses doucement et simplement. C’est vraiment fait à la fois pour ceux qui veulent des connaissances basiques à propos d’une instruction ou d’une fonction, mais aussi pour ceux qui veulent des informations compréhensibles et détaillées. Quelque soit votre niveau, vous serez parfaitement formé ☺ . Donc si vous utilisez le HOTAS Cougar pour la première fois, nous vous suggérons fortement de lire les premières sections de ce manuel, ou vous ne serez jamais capable d’exploiter le HOTAS Cougar à son maximum. Le HOTAS propose vraiment un système de programmation très souple qui vous laisse plusieurs possibilités pour arriver au même résultat, mais attention, si néanmoins cela se résume à programmer, et pour cette raison, plus vous serez méthodique et logique, plus vous arriverez à vous débrouiller.

Sans plus d’embarras, et bien, c’est à vous maintenant ! bonne lecture, utilisez le meilleur de ce controlleur ultime et montrez à ces détracteurs ce dont le HOTAS Cougar est capable!

Pour une traduction Espagnole de ce manuel, visitez S.V.P:

http://www.escuadron111.com

Pour une traduction Française de ce manuel, visitez S.V.P:

http://www.checksix-fr.com/

Traduction francaise par :
Guillaume "Ghostrider" Houdayer
Patrice "Skypat" Basquin
Eric "lascar Noir" Le Guyot
Sebastien "Pépé" Peeren

Pour une traduction Hollandaise de ce manuel, visitez S.V.P:
http://thrustmaster.vanree.net/

Pour une traduction Allemande de ce manuel, visitez S.V.P:
http://www.thrustmaster-x-files.de/

Pour une traduction Russe de ce manuel, visitez S.V.P:
http://www.hotas.ru

REMERCIEMENTS

Nos plus sincères remerciements vont aux personnes suivantes et aux sites internet pour toute leur aide et leur soutien aucours de ce grand projet, félicitations à tous! ☺

Beta testeurs

Olivier "Red Dog" Beaumont
Robin "Emacs" Breyl
Jan-Albert "Anvil" van Ree
James "Nutty" Hallows

Compagnies et Escadrille

Mark “Frugal” Bush & frugalsworld.com
Wingmen-alliance.com
Escuadron 111.com
Microsimulateur
SimHQ.com
Sim-arena.com
Aimsworth Coproration
Combatsim.com
Ubi Soft
Checksix-fr.com
Dogfighter.com
Desktopsims.com
Gamekult.com
Fast jet Flight Simulation (a.k.a. HAM technologies)

Merci à

Len "Viking1" Hjalmarson
Matt Wagner
James R Campisi
Flavien "Vox" Duhamel
François Pimenta
David "Micro" Vely
Philippe "Twech" Dezeure
Philippe "Jag" Dubois
Lew/+Silat
Rob Coppock
Laurent Espinasse
Fernando Oscar Garcia Minguillán
Guillaume "Ghostrider" Houdayer
Oleg Maddox
Jim Staud
Jean-Dominique "Bing" Belin
Emmanuel "Judy" Durant
Thomas "Doloop" Coulomb
Denis "Dugin" Blary
David "Zip" Pierron
Ulf Muckel
Hal Bryman
Jose "Oso" Benito
Stanislav "huMMer" Vartanian

THRUSTMASTER



HOTAS COUGAR

(PCC) PANNEAU DE CONFIGURATION
MANUEL DE REFERENCE

INTRODUCTION 13
PROFILS POUR LES AXES 14
- **DEFAULT (DEFAUT)** 14
- **SAVE (SAUVEGARDER)** 14
- **LOAD (CHARGER)** 14
- **DELETE (SUPPRIMER)** 14

MODES JOYSTICK 15
- **REACTION DES AXES** 15

SECTION PARAMETRES DES AXES 16
- **BOUTON DE PARAMETRAGE DES AXES** 16
  - Boutton Apply (Appliquer) 16
  - Bouton Retrieve (Récuperer) 16
- **ONGLET SETUP TAB (REGLAGES DES AXES)** 17
  - Changer le Réglage des Axes 17
  - Inverser l'action d'un Axe 20
  - Verrouiller un axe 20
  - Changer les Axes reconnus par Windows 21
- **AXES EN "REGLAGE PHYSIQUE"** 22
- **AXES EN "ACTIVER LES ETATS DES AXES WINDOWS"** 23
- **REGLAGES DES AXES** 23
  - Informations sur la Zone Floue 24
  - Calibration du Centre 26
  - Trim sur les Axes 26
  - Réglage de courbe 28
- **ONGLET STARTUP & CALIBRATION** 30
  - Options de démarrage (Startup Options) 30
  - Calibration 31
  - Calibration manuelle 33

ACTIONS ET AUTRES OPTIONS 34
- **RESTART DEVICE (REDEMARRER PERIPHERIQUE)** 34
- **BUTTON & AXIS EMULATION** 34
- **DOWNLOAD TO DEVICE** 34
- **POLL DEVICE** 36
- **HIDE / TASKBAR ICON FUNCTIONALITY** 36

1. CE QUE NOUS AVONS POUR VOUS ! 43
- **1.1 INTRODUCTION** 43
- **1.2 CONFIGURATION DE VOTRE JOYSTICK** 43
- **1.3 PREMIER CONTACT AVEC VOTRE GUIDE DE REFERENCE :** 44

2. LES PRINCIPES FONDAMENTAUX 46

2.1 LES PRINCIPES DE BASE DE LA PROGRAMMATION THRUSTMASTER 46
2.1.1 Introduction....46
2.1.2 LE concept du HOTAS....46
2.1.3 Comment avons nous conçu le HOTAS pour les simulateurs et les jeux ?....47
2.1.4 Le fichier “joystick file” – Principes de programmation....47
2.1.5 Macros et « macro file »– Principes de programmation....48
2.1.6 Comment le fichier « joystick file » sait-il quel fichier « macro file » contient ses macros ? 49
2.1.7 Résumons ce que nous avons appris plus haut…....50
2.1.8 Charger le fichier joystick sur le système de vol....51
2.1.9 Structure des fichiers « joystick et macro files »....52
3. DECLARATION DES BOUTONS ET MACROS....54
3.1 DECLARATION DES BOUTONS ET SYNTAXE DU LANGAGE TM....54
3.2 SYNTAXE DES COMMANDES CLAVIER THRUSTMASTER....57
3.3 LES MACROS ET LES COMMANDES MACRO....58
3.4 MODIFIER UNE DECLARATION....60
3.5 LES ATTRIBUTS....61
3.5.1 Augmentation du nombre de positions programmables :....62
3.5.2 Séparation des macros sur un bouton :....65
3.5.3 Répétition ou non répétition des caractères :....70
3.5.4 Règles des barres d’attribut et hiérarchie....73
3.6 LES DECLARATIONS DE DELAI ET DE REPETITION....74
3.6.1 Déclaration DLY( )....74
3.6.2 Déclaration RPT( )....75
3.7 REGROUPEMENT DE CARACTERES – UTILISATION DES PARENTHESES....77
3.7.1 Les parenthèses ( )....77
3.7.2 Les parenthèses « crochet » { } (*Curly brackets)*....78
33.7.3 Les parenthèses obliques <> (*Angle brackets*)....79
3.8 UTILISATION ET DEFINITION DES BOUTONS DIRECTX....81
3.8.1 USE ALL_DIRECTX_BUTTONS....83
3.9 UTILISATION DES CODES KD, KU ET USB....85
3.9.1 KD, KU....85
3.9.2 Programmation USB....86
4. PROGRAMMATION DES HAT....87
4.1 PROGRAMMATION DES JOYSTICK HAT....87
4.1.1 Positions programmables sur un hat....87
4.1.2 Chapeau 4-voies contre 8-voies : USE HatID FORCED_CORNERS....88
4.1.3 Contrôler la souris avec un HAT.....89
4.1.4 Paramétrer un HAT comme Point de Vue (POV) HAT....91
4.1.5 Utiliser un HAT pour émuler les touches flèches du clavier....92

4.1.6 Utiliser un HAT pour émuler le pave numérique 92
4.1.7 Comment le Compilateur transforme l'instruction USE HatID AS 94

5. INSTRUCTIONS DE CONFIGURATION 97

5.1 INTRODUCTION 97
5.2 MDEF – FICHIER DE DEFINITIONS DE MACROS 98
5.3 REPETITION 99
5.4 S3_LOCK ET S3_UNLOCK 100
5.5 ATTRIBUTION D'UN AUTRE BOUTON POUR /I, /O AVEC SHIFTBTN 101
5.6 SENSIBILITE DU «HAT» – USE HAT SENSITIVITY 101
5.7 UTILISER LA SENSIBILITE DU BOUTON T1 102
5.8 USE FOXY GRAPHIC ET README 103
5.9 NULLCHR – CARACTERE NUL ^ 103
5.10 CLAVIER - KEYBOARD (AZERTY, QWERTY) 105
5.11 UTILISATION DES PROFILS A PARTIR DU 106
PANNEAU DE CONTROLE COUGAR - USE PROFILE 106
5.11.1 Informations complémentaires de Profil 106
5.12 INSTRUCTIONS DE CONFIGURATION DECRITES AILLEURS DANS CET OUVRAGE DE REFERENCE 108

6. PROGRAMMATION DES AXES 109

6.1 PRINCIPES DE BASE 109
6.1.1 Différences entre analogique et numérique 109
6.1.2 Les axes du Cougar 110
6.2 INSTRUCTIONS DE TYPE DIGITAL 111
6.2.1 Type 1: Répétition de caractères 111
6.2.2 Type 2 : Séquence normale de caractères, régions définies 117
6.2.3 Type 3: Génération continue de caractères 119
6.2.4 Type 4: Génération repetee de caractères 120
6.2.5 Type 5: Sequences programmees de caractères , Zones variables 121
6.2.6 Type 6: Génération répétée de caractères , zones variables 122
6.2.7 Sens des axes, valeurs analogiques et déclaration s digitales. 124
6.3 COURBES DE REPONSE (CURVE) 130
6.4 TRIM D AXES (TRIM) 133
6.5 DESACTIVATION D'AXES 137
6.5.1 Activation et désactivation d axes en vol avec LOCK, UNLOCK 138
6.6 AFFECTATION DES AXES (SWAP) 140
6.7 INVERSER LA DIRECTION D'UN AXE (REVERSE, FORWARD) 142
6.8 LA DECLARATION : USE AXES_CONFIG 143

7. PROGRAMMATION DE LA SOURIS 145

7.1 COMPRENDRE LA SOURIS ET LE MICROSSICK 146

7.2 USE MTYPE, LA FAÇON LA PLUS SIMPLE D'ASSIGNER LA SOURIS AU MICROSTICK. 147
7.3 UTILISER LE MICROSTICK COMME SOURIS....... 149
7.3.1 Assigner les autres axes aux axes de la souris 152
7.4 CREER UNE SOURIS PERSONNALISEE SUR LE MICROSTICK 154
7.5 USE ZERO_MOUSE 160
7.6 PROGRAMMER LES BOUTONS SOURIS 160
7.7 DESACTIVER L'ASSIGNEMENT PAR DEFAUT DE LA SOURIS AU MICROSTICK 161
7.8 INSTRUCTIONS MOUVEMENTS AVANCES SOURIS....... 162
7.8.1 Définir la résolution d'écran....... 162
7.8.2 Aller à une position spécifique de l'écran....... 163
7.8.3 Mouvement relatif à la position courante de la souris....... 164
7.8.4 Mouvement Circulaire/Polygonal....... 166

8. PROGRAMMATION LOGIQUE....... 170

8.1 PROGRAMMATION LOGIQUE – LES BASES....... 170
8.1.1 Comprendre les flags 170
8.2 DEFINIR LES FLAGS LOGIQUES ET LEURS INSTRUCTIONS BOUTON 171
8.3 OPERATEURS LOGIQUES....... 173
8.4 LA BASCULE LOGIQUE 174
8.5 UTILISER LES FONCTIONS LOGIQUES DELAY ET PULSE....... 176
8.5.1 La fonction Delay....... 176
8.5.2 La fonction Pulse....... 177
8.5 EXEMPLES DE PROGRAMMATION LOGIQUE....... 178
8.5.1 Alterner une instruction de type 4 entre ON et OFF....... 178
8.5.2 Une fonction de trim 'lent'....... 178

9. DEPANNAGES....... 179

9.1 RESET DES CONTROLLEURS....... 179
9.1.1 Encours de jeu: EMPTY_BUFFERS et STICK_OFF 180
9.1.2 Sous Windows....... 181

10. ANNEXES....... 183

ANNEXE 1. RESUME DES INSTRUCTIONS THRUSMASTER....... 183
Instructions Boutons et attributs instructions....... 183
Attributs Slash et attributs instructions 184
Instructions de configuration 185
Programmation des axes 186
Instructions avancées Souris 186
Instructions logiques....... 187
Instructions Matérielles....... 187
APPENDICE 2. SYNTAXE TOUCHE THRUSTMASTER 188
ANNEXE 3. CODE USB MAJUSCULE ET MINUSCULE....... 189

APPENDIX 4. DIFFERENCES ENTRE LES ANCIENS FICHIERS TM ET LES FICHIERS COUGAR 193
1. Changements dans la syntaxe des touches 193
2. Changements des attributs '/' 194
3. Instructions désormais non supportées 194
4. Extension de fichier, noms de fichier 194
5. Actions par défaut 195
6. Axes digitales vc axes analogiques. 195
7. Instructions digitales Type 1 196
8. Manette des gaz non présente 196
9. Macros – caractères interdits 196
10. RPT 196
11. le caractère de commentaire // 197

Introduction

Le HOTAS Cougar est un joystick ne nécéssitant pas de driver et qui ne requiert aucun programme tournant en tâche de fond pour utiliser differentes configurations sur les axes ou pour exécuter une émulation quelle qu'elle soit. Pour cette raison il est important que le seul le joystick soit au courant des changements appliqués sur ses axes, donc, les autres applications telles CTFJ (utilitaire central pour joystick de Bob Church) ou encore l'utilitaire de calibration Windows ne doivent pas être utilisés, si c'etait le cas, plusieurs changements avancés sur la calibration des axes pourrait avoir des résultats divergeants.

<u>NOTE IMPORTANTE</u>: N'UTILISEZ PAS LA CALIBRATION WINDOWS POUR REGLER L'HOTAS COUGAR, UTILISEZ LE PANNEAU DE CONTROLE DU HOTAS COUGAR POUR LE CALIBRER MANUELLEMENT.

L' application Panneau de Contrôle du HOTAS Cougar (PCC) est utilisé pour changer diverss paramètres en sortie du HOTAS Cougar, cela va de l'affectation des axes, aux diffèrents modes de sortie du joystick (mode emulation, mode Windows…). Ce qui va suivre dans ce manuel de référence est une explication bouton après bouton de l'application, et par quelle manière les changements apportés sur ces boutons changeront le comportement du Joystick.

Pour afficher les réglages qui sont actuellement actifs dans le Joystick, ouvrez le PCC HOTAS tout en ayant le Joystick connecté.

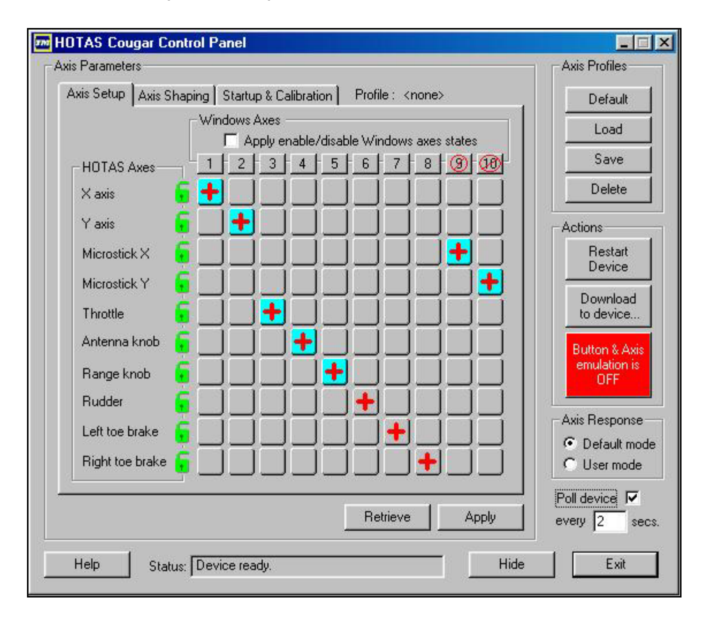


Figure 1: Panneau de contrôle du HOTAS Cougar en configuration par défaut

Profils pour les Axes

Les profils sont utilisés pour charger rapidement des configurations pré-définis dans le PCC HOTAS et dans le joystick. Dans la partie Profiles du PCC HOTAS, Quatre boutons sont accessibles: "Default", "Load", "Save" and "Delete" , chacun d'entre eux est expliqué dans la section ci-dessous.

Default (défaut)

En cliquant sur sur le bouton Default, on affiche les paramètres par Défaut utilisés par le joystick en mode Windows. Cela inclus l'affectation de tous les axes selon les standards DirectX, la plupart des axes désignent des directions, la zone floue haut est bas est de 5%, la zone flou centre est de 7%, la courbe linéraire de réponse (0), la base de la courbe est centrée, et le bouton 'apply enable/disable Windows axes states" est désactivé. Si le joystick est connecté et que les réglages affichés n'ont aucun sens, la meilleure solution est alors de cliquer sur le bouton Default, et de repartir sur ces réglages pour les changements.

Save (Sauvegarder)

En cliquant sur le bouton Save, on sauvegarde dans le répertoire HOTAS/profiles la configuration actuelle, qui a été definie dans la partie concernant les Paramètres d'Axes, en utilisant une extension de fichier '.TMC' (Configuration Thrustmaster). Si une quelconque Calibration a été effectuée, elle sera aussi enregistrée dans ce fichier. Sauvegarder les profils de cette manière permet de sauvegarder plusieurs paramètres d'axes pour diffèrents jeux, et peut être chargé pour de futures utilisations.

Load (Charger)

En cliquant sur le bouton Load, on charge une configuration (profil) dans l'ecran de contrôle(Le tableau). Cela ne signifie pas que la configuration est dans le Joystick, pour que ce soit la cas, vous devez toujours cliquer sur le bouton Apply comme décrit plus loin dans ce manuel de référence.

Delete (Supprimer)

En cliquant sur ce bouton, on supprime un profil.

Modes Joystick

Le PCC du HOTAS vous permet de régler les modes du Joystick se rapportant aux Paramètres d'Axes, aux Options de Calibration et les fontionnalités d'Emulation. Régler ces modes du Joystick ne nécéssite pas de cliquer sur le bouton Apply car chaque changement sur la réaction de l'Axe entrainera un changement de mode du Joystick.

Reaction des axes

Dans le mode par défaut, le joystick utilisera le réglage par défaut des axes et les données pre-formattées. Ces données sont toujours chargés dans le Joystick, et sont utilisés toutes les fois ou le joystick est connecté, il faut encore que le status visible ai été spécifié lors du derrnier transfert (ceci est explique par la suite dans la section “Changer les etats des axes Windows”). Quand la réaction de l'axe est réglé à User Defined Mode, le joystick utilisera les données suivantes que vous aurez spécifiées:

- Axis Mapping (Affectation des axes)
- Reversing Data
- Locking Data (Verrouiller les données)
- Curve Information (informations sur la courbe)
- Base of Curve (base de la courbe)
- Dead Zone Information (Information sur la zone floue)
- Trim Settings (Reglages Trim)

Pour des informations concernant les options de Calibration, consultez la section Calibration. Pour des informations concernant les modes d'Emulations, consultez la section Emulation des Boutons & Axes.

Section Paramètres des Axes

L’application CPP du HOTAS est constituée de trois onglets : L’onglet Axis Setup (Réglage des axes), l’onglet Axis Shaping (Mise en forme des axes) et l’onglet Startup & Calibration (Démarrage et Calibration). En fait, exécuter une ou plusieurs changements sur ces trois onglets n’entrainera pas de changement immédiat sur le Joystick, et l’action des axes devra être réglé sur User mode avant que le joystick puisse utiliser cette nouvelle information. Pour mieux suivre les explications données, d’abord ouvrez le PCC du HOTAS, et cliquez sur le bouton Defaut profile

Bouton de Paramétrage des Axes

La section de Paramétrage des Axes (hormis l’axe nommé ci-dessus) contient deux boutons : “Retrieve” et “Apply”. Leurs fonctionnalités et leur utilisation sont expliquées dans la section suivante.

Boutton Apply (Appliquer)

Le bouton Apply permet de transferer les paramètres de configuration décrit dans la partie Paramétrage des Axes vers le Joystick. Jusqu’à ce que vous exécutiez cette opération, le Joystick n’a aucune connaissance des changements appliqués dans le paramétrage des axes, et le bouton Apply transférera n’importe quel paramètre qui est actuellement affiché ou défini dans le Tableau onglet. Pour que le Joystick utilise les informations transférés, l’action des Axes doit-être réglée sur user Mode, qui est automatiquement enclenché en cliquant sur le bouton Apply.

Bouton Retrieve (Récuperer)

Le bouton Retrieve est utlisé pour récupérer la configuration actuelle sauvegardée dans le joystick. Si un des paramètres a été changé en mode emulation, cela permet de les restituer en lancant une opération de récupération, et en vérifiant la partie appropriée. Le PCC du HOTAS exéxute automatiquement une opération de Récupération au démarrage pour afficher la configuration actuelle; si le Joystick n’est pas connecté, un message d’erreur est affiché, et le profil par Défaut est affiché. En cliquant sur le boutoin Retrieve, vous pouvez aussi voir le mode courant du joystick.

Onglet Setup Tab (réglages des axes)

Plusieurs fonctions sont disponible sur cet onglet, et leur objectif respectif peut-être résumé de la façon suivante.:

- Changer le Réglage des Axes
- Inverser l'action d'un Axe
- Verrouiller un axe
- Changer les Axes reconnus par Windows

Ces fonctions sont expliquées en détail dans les sections suivantes.

Changer le Réglage des Axes

Si vous avez besoin qu'un axe particulier contrôle un axe diffèrent dans une configuration spécifique, la manière la plus simple est d'utiliser le réglage des Axes. Quand il est chargé dans son format par défaut avec seulement le TQS (Manette des gaz) connecté, le réglage des Axes s'affichera de cette façon::

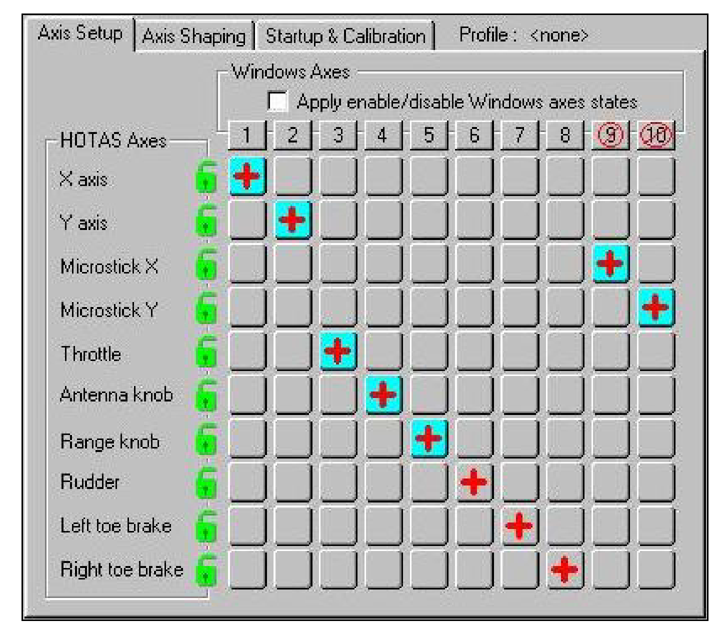


Figure 2: Onglet Axis Setup dans une configuration par defaut

Les boutons numérotés réprésentés au centre du tableau d'affectation des axes montre la configuration actuelle d'affectation; cela nous montre que l'axe X est affecté au prrmier axe, Y est affecté au second , la molette d'antenne est affecté au troisième
, etc… Si nous voulons que le second axe (axe Y DirectX) spoit controllé par la manette de gaz (ce qui est très utile pour les jeux de course), nous n'avons qu'à cliquer sur la ligne de la manette des gaz dans la seconde colonne, et la fenêtre montrera les changements suivants:

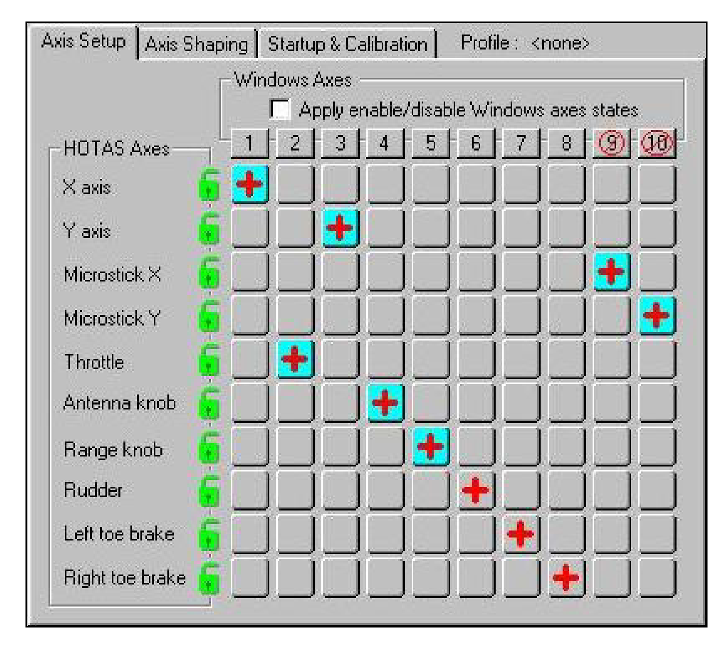


Figure 3: Tableau réglage des axes avec la manette des gaz utilisée comme Axe Y

Comme on peut le voir, la manette des gaz a été affectée à la seconde position,et l'axe Y a pris la place de la molette de la manette des gaz sur le troisième axe. Toutes les echanges d'axes peuvent être exécutés de cette façon, et bien que la fenêtre affichera les axes dans un état changé, tous les axes programmés à l'intérieur du Joystick, incluant toutes les inversions, les courbes, les zones floues, et les émulations programmées resteront spécifiques aux axes physiques; en conséquence si la manette des gaz est programmée pour avoir une courbe spécifique et si elle est echangée avec l'axe Y, la manette des gaz controlera l'axe Y de la même façon qu'elle controllais l'axe précedant.

Notez que le fond des boutons cochés apparaît en gris sur la sixième, la septième et la huitième colonne, contrairement aux autres (en bleu clair).

C’est parce que dans cette configuration, le RCS (le pallonier et les freins) ne sont pas connectés. Les boutons avec le fond bleu clair indiquent les axes qui sont physiquement connectés. Pour avoir les axes Y et Y controllés par le Micro-joystick, le réglage des axes pourrait être comme cela:

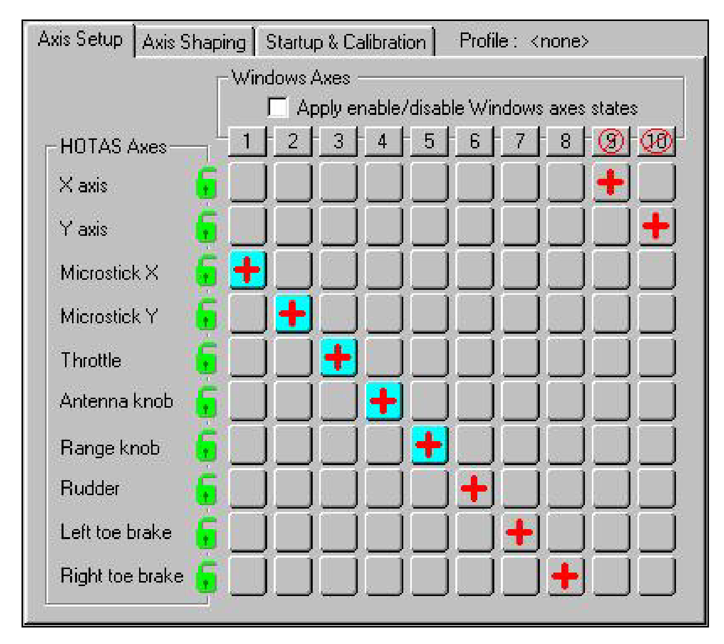


Figure 4: Tableau de réglage des Axes avec le micro-joystick controllant les axes X & Y

Inverser l'action d'un Axe

Pour inverser l'action d'un axe, cliquez sur le bouton désiré avec un signe +(ou signe -) et il alternera l'action d'un axe. Dans le cas ou vous voudriez changer l'action de l'axe Y, l'apparence du tableau de réglage des axes devraient être comme cela:

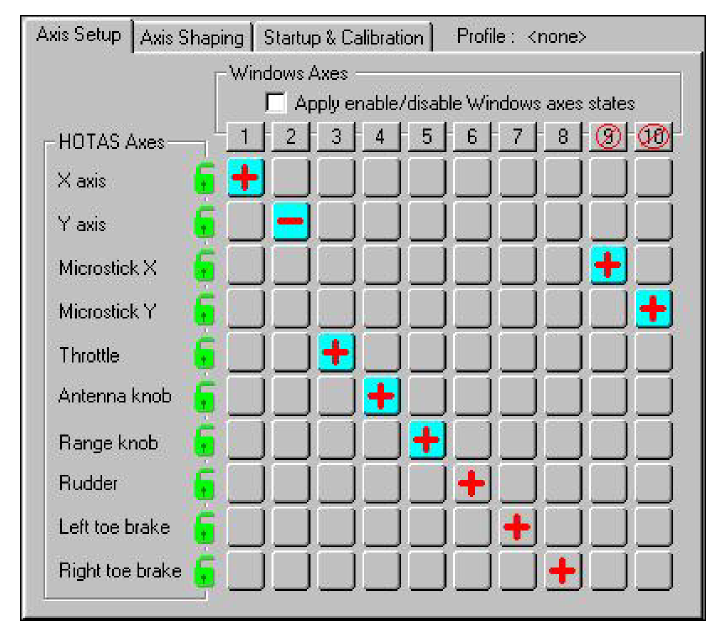


Figure 5: Axis Setup Tab with Y Axis reversed

Verrouiller un axe

Sur le coté droit de chacun des noms d'axe, il y a une icône en forme de cadenas. Ss ce cadenas est "ouvert" et que sa couleur est verte, l'axe est "dévérouillé" et se comportera normalement. Si le cadenas est fermé et que sa couleur est rouge, l'axe est "verouillé" et ses valeurs ne pourront être changés. Pour alterner l'etat du cadenas, cliquez sur lui, et il bougera d'un etat à un autre comme décrit ci-dessus.

Changer les Axes reconnus par Windows

Ce titre peut ne pas refleter precisemment le sujet abordé, mais sans renter dans des spécifications trop techniques, c'est la meilleure description de ce qui va suivre. Notons qu'un bouton est affiché en haut de chaque colonne, et dans le mode par défaut, les boutons 9 et 10 ont un cercle d'interdiction rouge en surimpression du texte, ces boutons sont representés ci-dessous:



Figure 6: Les "boutons numérotant les axes" situés sur le tableau de réglage des Axes

Chacun de ces boutons représente un axe DirectX diffèrent, ces axes sont:

Numéro Axe	Nom DirectX de l'AXE
1	Axe X
2	Axe Y
3	Axe Z
4	Axe X rotatif
5	Défilement 0
6	Axe Z rotatif
7	Défilement 1
8	Axe Z rotatif
9	<non disponible>
10	<non disponible>

Table 1: Numéro d'axes et noms DirectX associés

En plus du fait que vous ne pouvez pas utilisez le neuvième et le dixième axe, il est aussi impossible de désactiver le premier et le second axe, car la programmation des drivers du joystick implique qu'il y ait au moins deux axes. Si vous liquez sur l'un des boutons numérotés entre 3 et 8, le cercle d'interdiction rouge basculera entre les états ON (accitf) et OFF(Inactif). Quand un joystick est connecté, le nom des axes ainsi que le nombre d'axes "reconnus" par Windows, sont determinés par une ou deux sources possibles

1. Le réglage physique actuel du joystick
2. L'etat de ces 'boutons numérotés"

Les explications sur ces deux points figurent sur la page suivante.

Axes en "Réglage Physique"

Que voulons-nous dire par "Réglage Physique" ? Tout cela signifie quels sont les autres controlleurs (Manette de gaz, pallonier) qui sont connectés au joystick et qui définissent donc le nombre d'axes du Cougar détectés par Windows. 6 Réglages physiques sont disponibles pour le HOTAS..

1. Le Joystick est le seul connecté
2. Le Joystick est connecté avec le TQS
3. Le Joystick est connecté avec le RCS modèle 1 axe
4. Le Joystick est connecté avec le nouveau RCS modèle 3 axes
5. Le Joystick est connecté avec le TQS et avec le RCS modèle 1 axe
6. Le Joystick est connecté avec le TQS et avec le RCS modèle 3 axes

Chacune de ces diffèrentes possibilités affichera diverses combinaisons d'axes pour le HOTAS au sein de Windows. Ci-dessous un tableau vous représente chacune des 6 possibilités listées ci-dessus et affiche quels axes seront disponibles pour chacune des configurations.

	NOM DES AXES							
Configuration	X	Y	Z	Rx	SL0	Rz	SL1	Ry
1	●	●						
2	●	●	●	●	●			
3	●	●				●		
4	●	●				●	●	●
5	●	●	●	●	●	●		
6	●	●	●	●	●	●	●	●

Axes en "Activer les états des Axes Windows"

En utilisant les boutons numérotés, vous pouvez changer les axes qui seront reconnus par Windows. La proédure pour activer l'utilisation de ces boutons est décrite si-dessous.

1. Selectionner l'axe désiré que vous voudriez être "reconnu" par Windows en changeant l'état de chaque bouton numéroté.(Activer ou désactiver) jusqu'à ce que la configuration souhaitée soit atteint
2. Assurez vous que la case à cocher "Apply enable/disable Windows axes states" est activé.
3. Transférer le fichier dans le Joystick en utilisant le bouton "Apply", puis cliquez "OK" pour exécuter le re-démarrage du joystick.
4. Désormais, le Joystick sera reconnu par Windows en utilisant les axes spécifiés.

Le principal dans tout cela est que même si vous ne possédez ni l'ancien pallonier ou ni le nouveau (avec les freins), vous pouvez faire "croire" à DirectX que l'un ou l'autre de ces controlleurs sont présents, et vous les emulez alors avec l'un des axes disponibles ou même à travers un fichier d'émulation.

Il est important de noter que le Joystick se configurera dans le mode utilisateur "Enable Windows Axes States" aussi longtemps que le dernier fichier transféré dans le joystick aura la case à cocher ""Apply enable/disable Windows axes states" activé".

Réglages des axes

L'onglet "Axis Shaping" permet de modifier beaucoup plus de fonctionnalités avancées du Joystick, et c'est très utile quand l'utilisateur souhaite personnaliser chaque axe avec ses préférences personnelles. La liste déroulante 'axe à régler (Axis to set) en haut de l'onglet selectionne l'axe désiré, parmis dix possibilités, sur lequel vous allez effectuer les réglages. Une fois qu'un paramètre a été changé, le changement est affiché sur le graphique situé à la droite de la liste des diffèrents paramètres. La mise à jourde l'affichage s'effectue à changement de l'une des catégories de paramètres, ouen cliquant le bouton Refresh (Raffraichir) situé en haut à droite de l'onglet. Les divers paramètres ajustables sont les suivants

- Zone Floue Haut (ZFH)
- Zone Floue Bas (ZFB)
- Zone Floue Centre (ZFC)
- Calibration Center
- Axe Trim
- Réglages de la courbe (facteur et base)

Les paramètres listés ici peuvent être activés en selectionnant les diffèrents mode de sortie du Joystick.

Informations sur la Zone Floue

En changeant les informations de la Zone Floue (ZFH est la Zone Floue Haute, ZFB est la Zone Floue Basse, ZFC est la Zone floue Centre), vous pouvez modifier les zones inactives de chaque axe. Les changements sucessifs seront reflétés dans la partie droite de l'ecran ou les paramètres sont listés, les zones floues sont surlignées en rouge. Toutes les valeurs maximales ne peuvent dépasser 100%, 100% représente 30% de la capacité de mouvement du Joystick. En exemple, la valeur par défault utilisé en interne dans le joystick représente 1% de chacune des zones floues, ce qui nous donne un total de 3% sur toute l'intervalle de l'axe. Ci-dessous une photo d'écran représente cet onglet.

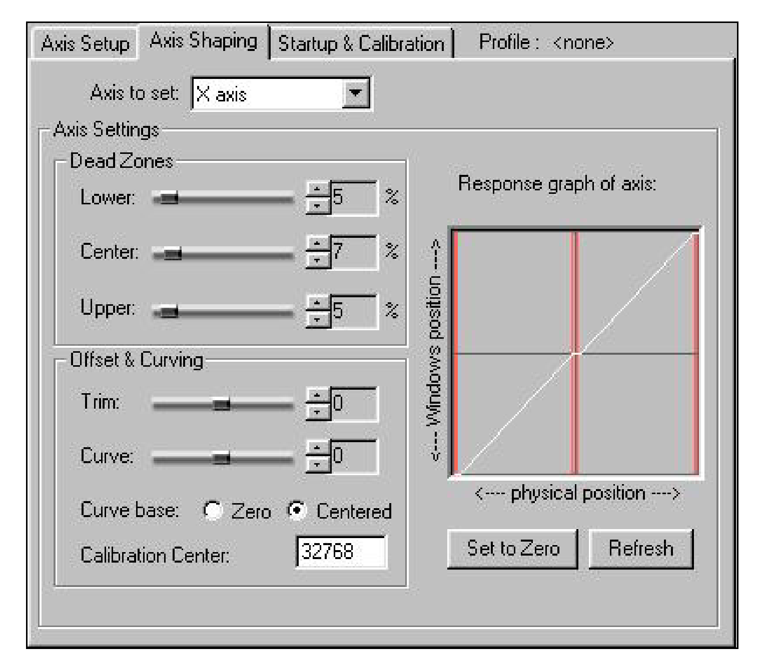


Figure 7: Onglet de Réglages des Axes avec les paramètres par défaut

Dans la figure ci-dessus, les petites parties en rouge sont apparentes à la fois sur les cotés et sur le centre. Ce tracé nous montre la relation entre la position physique des axes et les valeurs qui seront utilisés par Windows lors de l' utilisation du Joystick sur le bureau ou dans les jeux. Pour information, l'axe horizontal de ce graphique represente

les valeurs que le Joystick lis sur ces axes, avec la valeur minimum affichée sur le coté gauche du tracé. L'axe vertical du graphique répresente les données du Joystick 'lues' par Windows, avec la valeur minimum affichée en bas du tracé. On peut noter que dans la zone floue, le graphe devient une portion droite (horizontale) , voulant dire que la valeur de sortie sera la meme pour les differentes entrees. La zone floue bas est situé sur le coté droit du graphique, et il y aussi une portion de ligne droite(horizontale). La figure suivante illustre ce qui arrive si on augmente la valeur de la ZFC.:

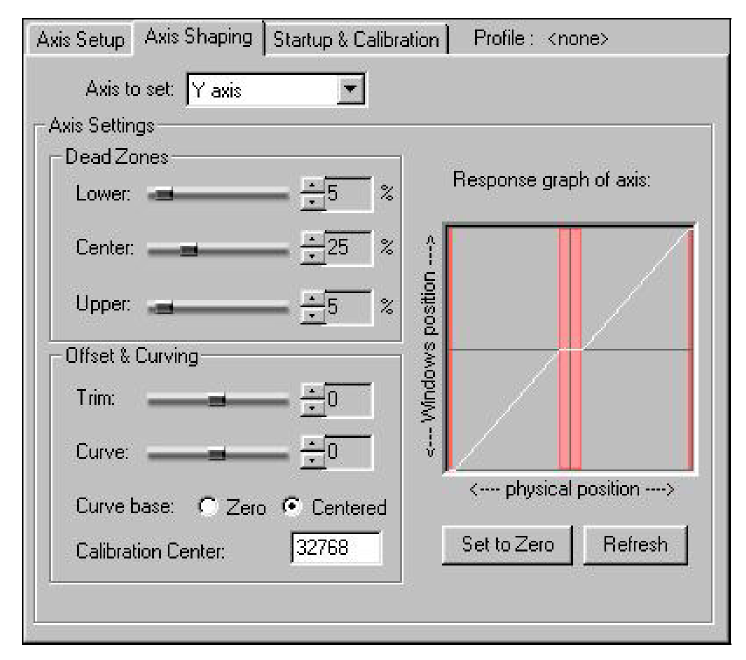


Figure 8: Onglet de Réglages des Axes avec augmentation de la valeur ZFC

La figure 8 montre clairement qu'augmenter la valeur de la Zone Floue Centre de 7% à 25% elargie proportionnelement la portion surlignée en rouge sur l'axe central horizontal, la ligne horizontale blanche représente la zone de l'axe ou le Joystick sera inactif. Les zones floues haute et basse réagissent de la même manière..

Calibration du Centre

Cette valeur représente la position centrale de l'axe du Joystick. Si la valeur de Position Centrale est inférieure à la valeur courante de la position physique centrale du joystick, alors l'axe atteindra une plus grande valeur quand le joystick sera au repos dans sa position centrale. Un résultat similaire peut-être constaté en ajustant la valeur de Trim pour compenser les valeurs physiques envoyés par le Joystick; Le principal avantage de la fonction Trim en plus de la valeur centrale et que le Trim peut-être modifié durant le mode emulation.

Trim sur les Axes

La fonction Trim est utilisée pour compenser les valeurs physiques des axes par de nouvelles valeurs. Par exemple, si l'axe du Joystick est à une position physique de 30% et que le Trim est réglé à -20%, alors la position actuelle lu par l'ordinateur sera de 10%. La valeur maximale du trim est de -/+ 50%; cela signifie que depuis la position centrale, il est possible d'avoir un axe allant d'une valeur à l'autre juste en ajustant le Trim. La figure suivante vous le démontre.

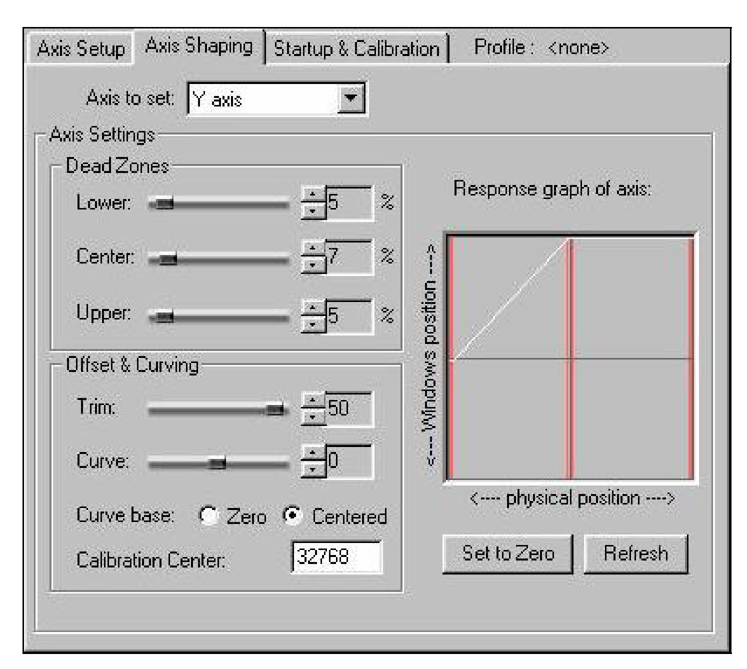


Figure 9: réglage du Trim au maximum

Avec le Trim appliqué à l'axe Y, la réaction peut-être observé immédiatement. Avec l'axe à sa position physique centrale, la donnée reçue par Windows représente la valeur maximale de l'axe Y. Si on bouge dans l'axe Y pour essayer d'augmenter cette valeur, il n'y aura aucune réaction; par contre si on bouge dans l'axe Y depuis sa position physique centrale vers sa position minimale, la valeur de position de l'axe diminue jusqu'à atteindre une valeur minimale qui est égale à la position normale centrale de l'axe (sans trim appliquée).

Si nous modifierions le Trim vers sa valeur minimale, le graphique ressemblerait à cela :

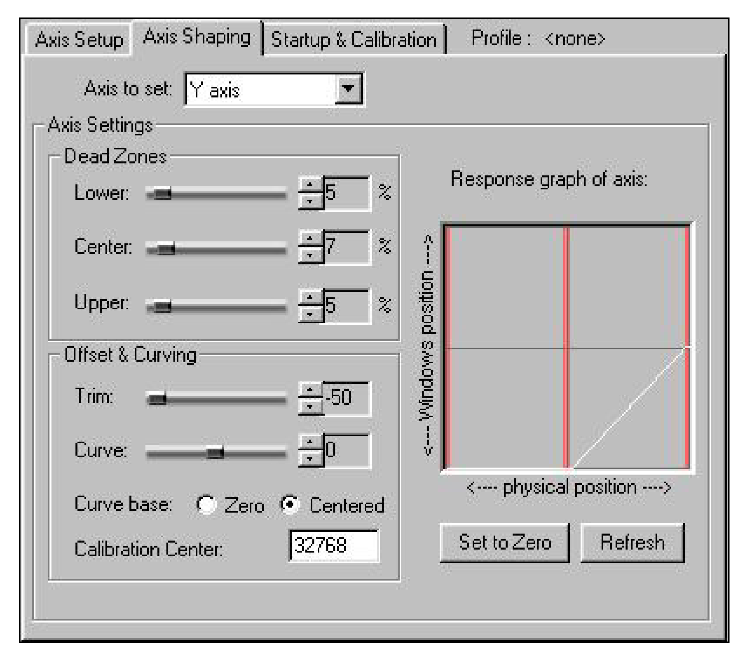


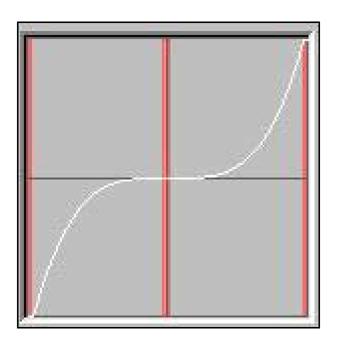
Figure 10: Trim à sa valeur minimale

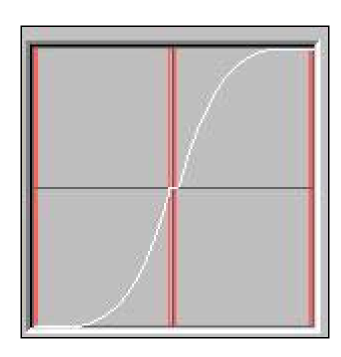
Avec ces changements appliqués, le valeur de l'axe Y serait à son minimum quand l'axe Y serait physiquement à sa position centrale, et serait à sa valeur centrale quand l'axe Y serait physiquement maintenue à sa position maximale.

Réglage de courbe

Ce réglage peut-être ajusté pour avoir une réponse de l'axe exponentielle plutôt qu'une réponse linéaire. Le paramètre de la courbe peut avoir une valeur comprise entre –32 et 32; toutefois, régler plusiseurs axes avec des valeurs hautes va ralentir enormémént le focntionnement du Joystick. Notons que des valeurs supérieures à 20 ou infèrieures à –20 ne sont pas très utiles, et que de tels changements au niveau de la courbe n'est pas suffisant pour justifier de telles valeurs.

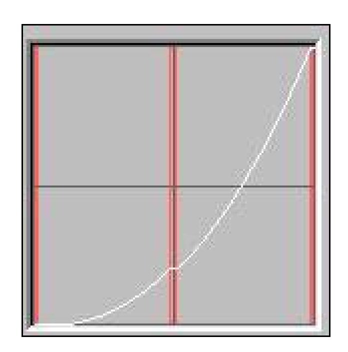
Nous allons tout d'abord étudier des courbes avec une base centrée. Alors qu'elle peut-être la diffèrence entre une courbe positive et une courbe négative ? regardons la réponse quand un réglage de 10, positif ou négatif est appliqué sur la courbe.

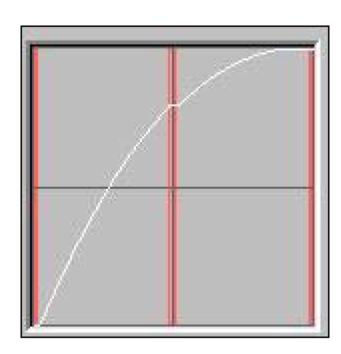




La figure à gauche représente l'axe avec un réglage de courbe de –10 (négatif); comme on peut le voir, autour de la position centrale, il y a très peu de changement sur l'axe presque comme si on augmentait la zone floue centrale. Le le taux de changement augmente rapidement jusqu'à ce que la pente de l'axe devienne presque verticale. Cela signifie que le Joystick offira une réponse très sensible aux alentours des limites de l'axe. La figure à droite montre le graphique d'un axe avec un réglage de courbe de 10, qui apparaît être presque être un réglage inverse; l'axe est beaucoup plus sensible autour de la position physique centrale, puisque qu'aux diffèrentes extrémités de l'axe, la sensibilité est de loin inférieure comme si on augmentait les zones floues haut et bas.

L'idée représentée par le radio bouton 'base de la courbe' est que malgrè le fait que les courbes de sensibilités du Joystick et du Microstick devraient être courbées, ou bien comme nous l'avons vu dans l'exemple précédant, une pente uniforme est préférable pour les axes tels les palonniers ou la manette de gaz. Ci-dessous des graphiques représentent des réglages de courbe positive et négative de 5 (10 est une valeur excessive), mais avec un réglage de la base de courbe à Zéro..





La figure de gauche représente le graphique pour un axe avec un réglage de Courbe de –5 (négatif) - Notons la similarité entre la section en haut à droite représentant un régale de courbe de 10 et base de courbe centrés. Le Joystick va réagir très lentement à l'extremité basse de l'axeThe Joystick will now react very slowly at the lower extreme of the axis, et sa sensibilité augmente alors jusqu'à ce qu'il atteigne un pic de valeur à l'extrémité haute de l'axe. A droite est représentée une Courbe avec un réglage positif de 5, et une base de zero. Une fois encore, notons la similarité entre le graphique complet de la courbe négative (base à 0) et la section en haut à droite de la courbe négative (base centrée). La sensibilité augmente prés du minimum physique de l'axe et diminue prés du maximum. L'extrémité haute de l'axe est comme si nous avions augmenté la zone floue haute.

Onglet Startup & Calibration

L'onglet "startup and calibration" (démarrage et calibration") contient des informations sur le comportement du Joystick au démarrage de l'ordinateur et sur la configuration de calibration. Cet onglet est séparé en 2 sections, la partie haute représente les options de démarrage et la partie basse, la configuration de calibration. Chacune des deux sections est décrite ci-dessous.

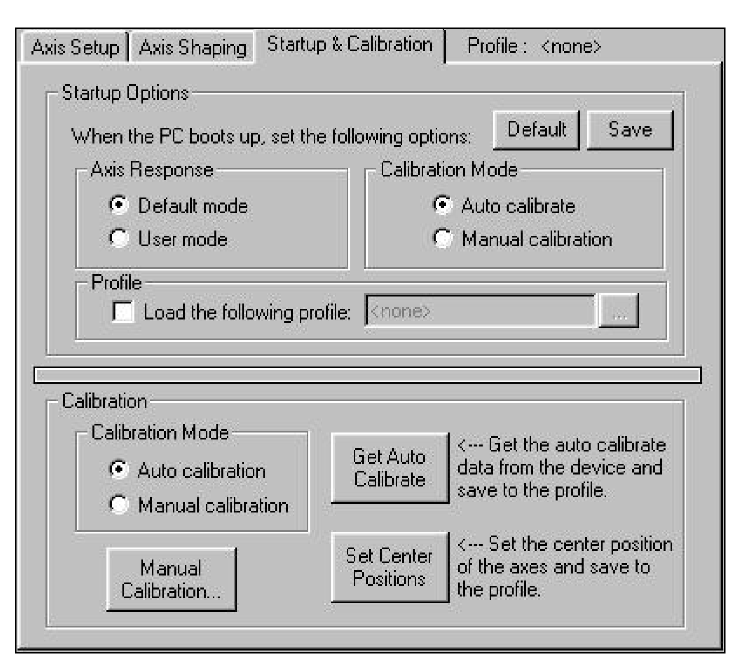


Figure 11: Onglet Startup & Calibration

Options de démarrage (Startup Options)

Il y a 3 options pouvant être prises en compte par le Joystick au démarrage de l'ordinateur. Ces options sont, les réponses des axes, le mode de calibration et le chargement d'un profil spécifique. Le Joystick par défaut utilisera un mode de réponse des axes par défaut, une calibration automatique et le dernier profil chargé dans le Joystick. Pour changer le comportemenrt par défaut, cliquer sur les options désirées et appuyer sur le bouton 'sauvegarder '('Save').. Pour choisir un profil cliquer sur la case à cocher puis cliquer sur le bouton '…'. Notons que le profil doit exister dans le sous-répertoire du répertoire du HOTAS. Pour récupérer les options de démarrage par défaut, cliquer sur le bouton 'défaut' ('default').

Calibration

La section de calibration contient les options pour choisir le mode de calibration, calibrer le Joystick, récupérer les données automatiques de calibration et régler les positions centrales des axes..

Choisir la calibration automatique (AUTO) ou manuelle (MANUAL) dans les options de mode de calibration . Notons qu'il n'est pas requis de cliquer sur le bouton 'Appliquer' ('Apply') pour appliquer le mode de calibration – c'est automatique.

En mode de calibration manuelle, les données utilisées seront celles qui ont été chargés dans le Joystick après une calibration manuelle. Ces données sont automatiquement chargés, avec les paramètres des axes, après toute calibration effectuée.

Le mode de calibration Auto va configurer les axes à leur position maximum comme si vous bougeriez un axe à son extrémité. La calibration manuelle signifie que le joystick utilisera les données de calibration crées lors du processus de calibration manuelle, décrit plus loin dans cette section. Notons que le Joystick ne peut basculer en mode calibration manuelle si aucune calibration manuelle n'a été effectuée. Si le joystick est initialisé (par re-connection ou en cliquant sur le bouton) et si il est en mode calibration Auto, il est recommandé de bouger tous les axes du Cougar (Joystick, manette de gaz, range, Antenne, Microstick) à leur position maximale et minimale et en les maintenant environ 3 secondes. Cela permettra à la calibration de collecter les informations depuis les axes et de régler précisemment les axes de votre controlleur.

Cliquer sur le bouton "Get Auto Calibration" avant de basculer depuis le mode Auto de calibration vers le mode manuel copie les données de calibration automatique vers le profil courant. Vous pouvez alors appliquer et sauvegarder le profil courant, ainsi le Joystick utilisera ces données et n'essaiera jamais d'ajuster ses valeurs.

Le bouton "Set Center Positions" est utilisé pour sauvegarder une certaine position sur chacun des axes comme étant la position centrale. Si vous souhaitez avoir des positions centrales diffèrentes sur les axes du Joystick et du Range plutôt que celles données par la calibration automatique, alors vous pouvez appuyer sur ce bouton et spécifier l'axe désiré comme expliqué ci-dessous.

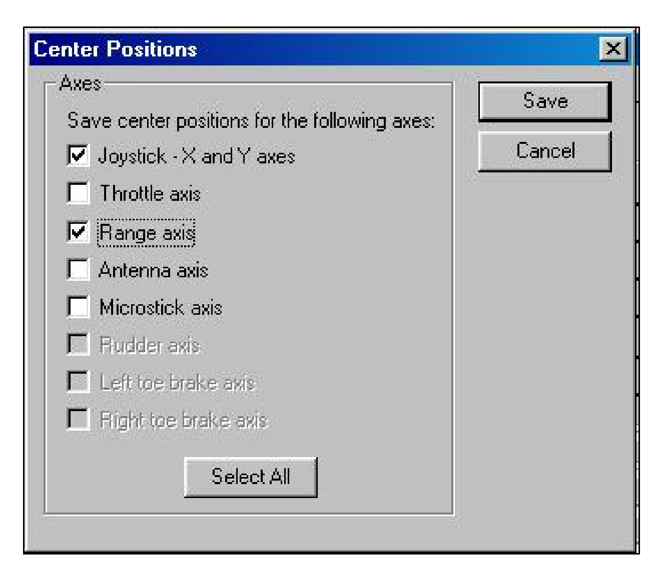


Figure 12: Sauvegarder la position centrale du Joystick et du Range

Une fois que vous pressez le bouton ‘Save’, on vous demande de bouger l’axe à la position centrale désirée et alors d’appuyer sur ‘OK’. Cela vous donnera alors la nouvelle position centrale de l’axe.

Calibration manuelle

Le bouton de calibration manuelle va vous entrainer vers la fenêtre de processus de calibration, comme ci-dessous:

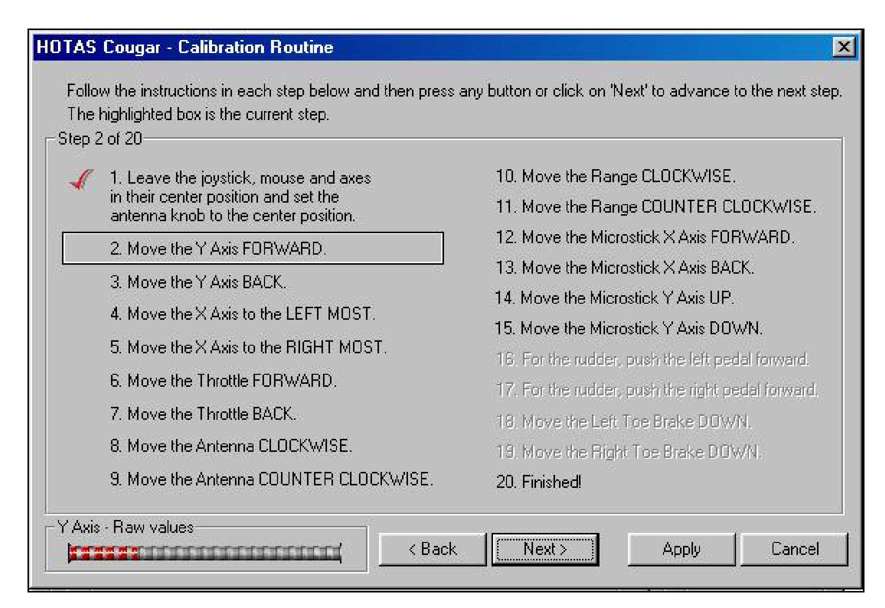


Figure 13: Fenêtre du processus de calibration manuelle

Suivez les directions données, pressez tous les boutons successivement, et cliquer sur le bouton ‘Next’ quand l’axe a atteint la position à sauvegarder pour l’étape en cours. Par exemple, si lors de la seconde étape, vous bougez l’axe Y vers la position avant et que vous cliquez sur le bouton ‘Next’, le Joystick sauvegardera la position de l’axe Y au moment même ou vous avez cliquer sur le bouton. Quand vous avez terminé le réglage de calibration, l’application va automatiquement envoyer les informations de calibration au Joystick, avec les données courantes chargés dans la section des paramètres axes. Les données de calibration sont enregistrées et peuvent être sauvegardées dans un fichier profil pour une utilisation future.

Notons que les axes non physiquement connectés sont grisés. Le processus de calibration va sauter la calibration des axes qui ne sont pas connectés. La barre de progression en bas à gauche de la fenêtre indique la valeur ‘raw’ de l’axe. – Il peut être impossible d’atteindre les valeurs maximum et minimum de la barre de progression. Cette dernière est utilisée comme un guide pour indiquer l’axe en mouvement et la direction.

Actions et autres options

Dans la section Actions, il y a trois boutons: 'Restart Device', 'Button & Axis emulation' et 'Download to device'. Il y aussi des options pour l'analyse automatique du joystick et le bouton 'Hide'. Toutes ses actions et options sont décrits ci-dessous.

Restart Device (rédémarrer périphérique)

Le bouton 'Restart Device' va entrainer une déconnection manuelle du Joystick, cela s'apparente à déconnecter et re-connecter le Joystick au port USB. Cette fonction est utile toutes les fois ou vous souhaitez changer les états des axes. Dans ce cas, il serait nécéssaire de déconnecter et de re-connecter le Josytick (comme expliqué dans la section "Changer le Réglage des Axes, ce qui est effectivement reproduit en cliquant sur le bouton 'Restart device'. Aussi, au démarrage, la calibration automatique mesure la position centrale des axes apropriés (X, Y, gouvernail, et Microstick); pour régler manuellement le centre des axes, cliquez sur le bouton 'restart' et maintenez les axes à la position désirée.

Button & Axis emulation

Si le bouton 'Button & Axis emulation' est ON (fond vert), alors le joystick utilisera le dernier fichier d'émulation chargé dans le joystick. Le fichier d'émulation est le fichier qui contrôle le clavier et la souris, comme pour les functionnalités concernant les diffèrents paramètres des axes, referez-vous au manuel de référence du propriétaire de HOTAS Cougar pour de plus amples informations.

Si le bouton 'Button & Axis emulation" est OFF (fond rouge), alors le joystick se comportera comme un joystick ordinaire avec des boutons fonctionnant comme les boutons DirectX de Windows.

Download to device

Le bouton 'Download to device' ouvre l'application 'Loader' du HOTAS Cougar.
L'application 'Loader' est utilisée pour charger les fichiers joystick (Fichiers joysticks Thrusmaster .TMJ). Pour de plus amples informations à propos de la structure des fichiers d'émulation, veuillez consulter le 'manuel de référence du propriétaire de HOTAS Cougar'. Sur la page suivante figure une image de l'application 'loader' du HOTAS Cougar.

Il y a deux boutons et une case à cocher sur la fenêtre principale du 'Loader' HOTAS Cougar. La case à cocher sert à détecter les erreurs des fichiers joysticks. Cela ne chargera pas le fichier vers le Joystick. Le bouton 'Download' vérifie le fichier joystick et le charge dans le Joystick si il n'y a pas d'erreur.

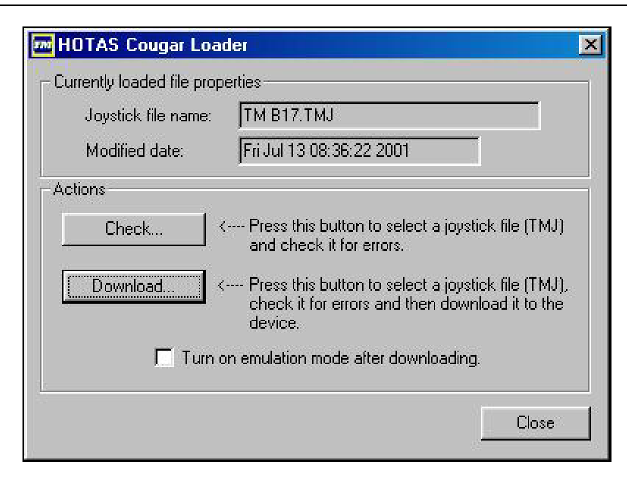


Figure 14: 'Loader' du Hotas Cougar pour charger les fichiers Joystick

'Currently loaded file properties' vous montre quel fichier est actuellement chargé dans le Joystick et la date à laquelle ce chargement a été effectué. La case à cocher 'Turn on emulation mode after downloading' est utilisée pour enclencher le mode emulation après qu'un fichier joystick ai été chargé dans le Joystick. Une fois que le fichier Joystick est chargé, le mode emulation doit-être enclenché afin que le Joystick puisse emuler les touches claviers pressés et les mouvements des axes enregistrés dans le fichier joystick. Cela peut-être aussi effectué en cliquant sur le bouton 'Button & Axis emulation button' dans le This can also be done by clicking the Button & Axis emulation button in the HOTAS PCC. Voir la section 'Button & Axis emulation' dans le dernier chapitre. Après verification ou chargement du fichier joystick, la fenêtre suivante apparaitra.

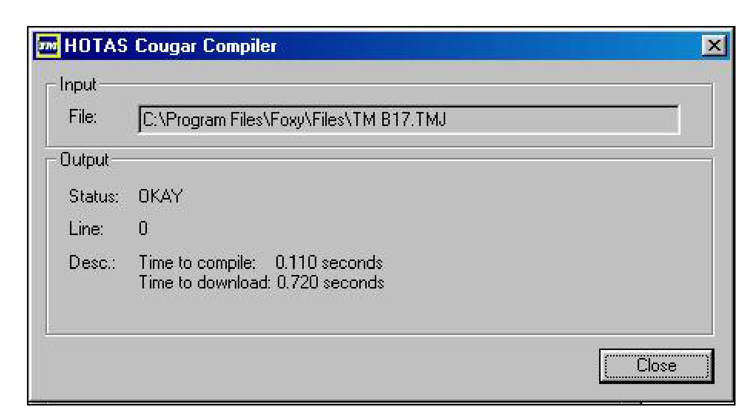


Figure 15: Compilateur du HOTAS Cougar pour vérifier les fichiers joysticks

Poll device

La case à cocher ‘polling device’ est utlisé par le PCC du HOTAS pour analyser l’etat actuel du Joystick. L’anlayse vérifiera si le joysticlk ets connecte et dans quel mode il se trouve. Le temps d’analyse est réglé dans le champ d’édition situé en dessous de la case à cocher.

Hide / Taskbar Icon functionality

Le PCC du HOTAS peut-être caché avec seulement un icône apparaissant dans la barre des tâches. Cliquer sur le bouton ‘Hide’ pour cacher le PCC du HOTAS. Notons que cette fonctionnalité peut-être utilisée seulement avec l’analyse enclenchée. Pour montrer le PCC du HOTAS après qu’il ai été caché, cliquer sur l’icône dans la barre des tâches avec le bouton droit de la souris et choisissez ‘Open HOTAS Cougar Control Panel…’ depuis le menu qui apparaitra. Les autres options disponibles depuis le menu permettent de sortir du PCC du HOTAS pour ouvrir le ‘Loader’ du HOTAS Cougar. Vous pouvez activer ou désactiver le mode emulation du Joystick en cliquant sur l’icône de la barre des tâches avec le bouton droit de la souris. La couleur de l’icône changera en accord avec la table suivante.

Icone	Code couleur	Description
	Avion vert / Fond jaune	Mode émulation **ON**, Réponse des axes en mode **utilisateur**
	Avion rouge / Fond jaune	Mode émulation **OFF**, Réponse des axes en mode **utilisateur**
	Avion vert / Fond gris	Mode émulation **ON**, Réponse des axes en mode **Windows**
	Avion rouge / Fond gris	Mode émulation **OFF**, Réponse des axes en mode **Windows**

Table 2: Description des icônes de la barre des tâches

THRUSTMASTER



HOTAS COUGAR

Guide de référence de l'acheteur de Cougar.

CONTENTS

INTRODUCTION 13
PROFILS POUR LES AXES 14
DEFAULT (DEFAUT) 14
SAVE (SAUVEGARDER) 14
LOAD (CHARGER) 14
DELETE (SUPPRIMER) 14
MODES JOYSTICK 15
REACTION DES AXES 15
SECTION PARAMETRES DES AXES 16
BOUTON DE PARAMETRAGE DES AXES 16
Boutton Apply (Appliquer) 16
Bouton Retrieve (Récuperer) 16
ONGLET SETUP TAB (REGLAGES DES AXES) 17
Changer le Réglage des Axes 17
Inverser l'action d'un Axe 20
Verrouiller un axe 20
Changer les Axes reconnus par Windows 21
Numéro Axe *21*
AXES EN "REGLAGE PHYSIQUE" 22
AXES EN "ACTIVER LES ETATS DES AXES WINDOWS" 23
REGLAGES DES AXES 23
Informations sur la Zone Floue 24
Calibration du Centre 26
Trim sur les Axes 26
Réglage de courbe 28
ONGLET STARTUP & CALIBRATION 30
Options de démarrage (Startup Options) 30
Calibration 31
Calibration manuelle 33
ACTIONS ET AUTRES OPTIONS 34
RESTART DEVICE (REDEMARRER PERIPHERIQUE) 34
BUTTON & AXIS EMULATION 34
DOWNLOAD TO DEVICE 34
POLL DEVICE 36
HIDE / TASKBAR ICON FUNCTIONALITY 36
1. CE QUE NOUS AVONS POUR VOUS ! 43
1.1 INTRODUCTION 43
1.2 CONFIGURATION DE VOTRE JOYSTICK 43

1.3 PREMIER CONTACT AVEC VOTRE GUIDE DE REFERENCE : ... 44

2. LES PRINCIPES FONDAMENTAUX ... 46

2.1 LES PRINCIPES DE BASE DE LA PROGRAMMATION THRUSTMASTER 46

2.1.1 Introduction ... 46
2.1.2 LE concept du HOTAS ... 46
2.1.3 Comment avons nous conçu le HOTAS pour les simulateurs et les jeux ? ... 47
2.1.4 Le fichier “joystick file” – Principes de programmation ... 47
2.1.5 Macros et « macro file »– Principes de programmation ... 48
2.1.6 Comment le fichier « joystick file » sait-il quel fichier « macro file » contient ses macros ? 49
2.1.7 Résumons ce que nous avons appris plus haut… ... 50
2.1.8 Charger le fichier joystick sur le système de vol ... 51
2.1.9 Structure des fichiers « joystick et macro files » ... 52

3. DECLARATION DES BOUTONS ET MACROS ... 54

3.1 DECLARATION DES BOUTONS ET SYNTAXE DU LANGAGE TM ... 54
3.2 SYNTAXE DES COMMANDES CLAVIER THRUSTMASTER ... 57
3.3 LES MACROS ET LES COMMANDES MACRO ... 58
3.4 MODIFIER UNE DECLARATION ... 60
3.5 LES ATTRIBUTS ... 61

3.5.1 Augmentation du nombre de positions programmables : ... 62
3.5.1.1 /U, /M, /D - Up, Middle, Down ... 63
3.5.1.2 /I, /O - In, Out ... 63
3.5.2 Séparation des macros sur un bouton : ... 65
3.5.2.1 /T – Barre d’attribut Toggle ... 65
3.5.2.2 Annulation de la mise en place d’une commande /T ... 67
3.5.2.3 Inverser la direction de la répétition ... 68
3.5.2.4 /P, /R – Appuyer et relâcher (Press et Release) ... 69
3.5.3 Répétition ou non répétition des caractères : ... 70
3.5.3.1 Non répétition des caractères ... 70
3.5.3.2 /A – Répétions automatique (Auto-Repeat) ... 71
3.5.3.3 /H –Maintenir (Hold) ... 71
3.5.4 Règles des barres d’attribut et hiérarchie ... 73
3.5.4.1 Règles des barres d’attribut ... 73
3.5.4.2 Hiérarchie des barres d’attribut ... 73

3.6 LES DECLARATIONS DE DELAI ET DE REPETITION ... 74

3.6.1 Déclaration DLY( ) ... 74
3.6.2 Déclaration RPT( ) ... 75

3.7 REGROUPEMENT DE CARACTERES – UTILISATION DES PARENTHESES ... 77

3.7.1 Les parenthèses ( ) ... 77
3.7.2 Les parenthèses « crochet » { } (*Curly brackets*) ... 78
33.7.3 Les parenthèses obliques <> (*Angle brackets*) ... 79

3.8 UTILISATION ET DEFINITION DES BOUTONS DIRECTX......................... 81
3.8.1 USE ALL_DIRECTX_BUTTONS.. 83
3.9 UTILISATION DES CODES KD, KU ET USB.. 85
3.9.1 KD, KU.. 85
3.9.2 Programmation USB 86
4. PROGRAMMATION DES HAT 87
4.1 PROGRAMMATION DES JOYSTICK HAT 87
4.1.1 Positions programmables sur un hat.. 87
4.1.2 Chapeau 4-voies contre 8-voies : USE HatID FORCED_CORNERS..................... 88
4.1.3 Contrôler la souris avec un HAT. 89
4.1.4 Paramétrer un HAT comme Point de Vue (POV) HAT 91
4.1.5 Utiliser un HAT pour émuler les touches flèches du clavier................................ 92
4.1.6 Utiliser un HAT pour émuler le pave numérique 92
4.1.7 Comment le Compilateur transforme l'instruction USE HatID AS........................ 94
5. INSTRUCTIONS DE CONFIGURATION.. 97
5.1 INTRODUCTION 97
5.2 MDEF – FICHIER DE DEFINITIONS DE MACROS... 98
5.3 REPETITION 99
5.4 S3_LOCK ET S3_UNLOCK... 100
5.5 ATTRIBUTION D'UN AUTRE BOUTON POUR /I, /O AVEC SHIFTBTN 101
5.6 SENSIBILITE DU «HAT» – USE HAT SENSITIVITY.. 101
5.7 UTILISER LA SENSIBILITE DU BOUTON T1... 102
5.8 USE FOXY GRAPHIC ET README... 103
5.9 NULLCHR – CARACTERE NUL ^ 103
5.10 CLAVIER - KEYBOARD (AZERTY, QWERTY).. 105
5.11 UTILISATION DES PROFILS A PARTIR DU 106
PANNEAU DE CONTROLE COUGAR - USE PROFILE..................................... 106
5.11.1 Informations complémentaires de Profil.. 106
5.12 INSTRUCTIONS DE CONFIGURATION DECRITES AILLEURS DANS CET OUVRAGE DE REFERENCE 108
6. PROGRAMMATION DES AXES... 109
6.1 PRINCIPES DE BASE.. 109
6.1.1 Différences entre analogique et numérique 109
6.1.2 Les axes du Cougar 110
6.2 INSTRUCTIONS DE TYPE DIGITAL 111
6.2.1 Type 1: Répétition de caractères 111
6.2.1.1 Comprendre le modificateur FORCE_MACROS... 113
6.2.1.2 Considérations important l'utilisation de FORCE_MACROS.......................... 114
6.2.2 Type 2 : Séquence normale de caractères, régions définies.............................. 117
6.2.2.1 Comprendre le modificateur - FORCE_MACROS 118
6.2.3 Type 3: Génération continue de caractères 119

6.2.4 Type 4: Génération repetee de caractères 120
6.2.5 Type 5: Sequences programmees de caractères , Zones variables 121
6.2.5.1 Comprendre l attribut FORCE_MACROS 122
6.2.6 Type 6: Génération répétée de caractères , zones variables 122
6.2.6.1 Comprendre l attribut FORCE_MACROS 123
6.2.7 Sens des axes, valeurs analogiques et déclaration s digitales. 124
6.2.7.1 Analogue axes values 124
6.2.7.2 Déclaration s digitales d axe de TYPE 1 125
6.2.7.3 Déclaration s digitales d axe de TYPE 2 125
6.2.7.4 Déclaration s digitales d axe de TYPE 3 126
6.2.7.5 Déclaration s digitales d axe de TYPE 4 127
6.2.7.6 Déclaration s digitales d axe de TYPE 5 127
6.2.7.7 Type 6 Digital axes statements 129
6.3 COURBES DE REPONSE (CURVE) 130
6.4 TRIM D AXES (TRIM) 133
6.5 DESACTIVATION D'AXES 137
6.5.1 Activation et désactivation d axes en vol avec LOCK, UNLOCK 138
6.6 AFFECTATION DES AXES (SWAP) 140
6.7 INVERSER LA DIRECTION D'UN AXE (REVERSE, FORWARD) 142
6.8 LA DECLARATION : USE AXES_CONFIG 143
7. PROGRAMMATION DE LA SOURIS 145
7.1 COMPRENDRE LA SOURIS ET LE MICROSSICK 146
7.2 USE MTYPE, LA FAÇON LA PLUS SIMPLE D'ASSIGNER LA SOURIS AU MICROSTICK. 147
7.3 UTILISER LE MICROSTICK COMME SOURIS 149
7.3.1 Assigner les autres axes aux axes de la souris 152
7.4 CREER UNE SOURIS PERSONNALISEE SUR LE MICROSTICK 154
7.5 USE ZERO_MOUSE 160
7.6 PROGRAMMER LES BOUTONS SOURIS 160
7.7 DESACTIVER L'ASSIGNEMENT PAR DEFAUT DE LA SOURIS AU MICROSTICK 161
7.8 INSTRUCTIONS MOUVEMENTS AVANCES SOURIS 162
7.8.1 Définir la résolution d'écran 162
7.8.2 Aller à une position spécifique de l'écran 163
7.8.3 Mouvement relatif à la position courante de la souris 164
7.8.4 Mouvement Circulaire/Polygonal 166
8. PROGRAMMATION LOGIQUE 170
8.1 PROGRAMMATION LOGIQUE – LES BASES 170
8.1.1 Comprendre les flags 170
8.2 DEFINIR LES FLAGS LOGIQUES ET LEURS INSTRUCTIONS BOUTON 171
8.3 OPERATEURS LOGIQUES 173
8.4 LA BASCULE LOGIQUE 174

8.5 UTILISER LES FONCTIONS LOGIQUES DELAY ET PULSE 176
8.5.1 La fonction Delay 176
8.5.2 La fonction Pulse 177
8.5 EXEMPLES DE PROGRAMMATION LOGIQUE 178
8.5.1 Alterner une instruction de type 4 entre ON et OFF 178
8.5.2 Une fonction de trim 'lent' 178
9. DEPANNAGES 179
9.1 RESET DES CONTROLLEURS 179
9.1.1 Encours de jeu: EMPTY_BUFFERS et STICK_OFF 180
9.1.2 Sous Windows 181
10. ANNEXES 183
ANNEXE 1. RESUME DES INSTRUCTIONS THRUSMASTER 183
Instructions Boutons et attributs instructions 183
Attributs Slash et attributs instructions 184
Instructions de configuration 185
Programmation des axes 186
Instructions avancées Souris 186
Instructions logiques 187
Instructions Matérielles 187
APPENDICE 2. SYNTAXE TOUCHE THRUSTMASTER 188
ANNEXE 3. CODE USB MAJUSCULE ET MINUSCULE 189
APPENDIX 4. DIFFERENCES ENTRE LES ANCIENS FICHIERS TM ET LES FICHIERS COUGAR 193
1. Changements dans la syntaxe des touches 193
2. Changements des attributs '/' 194
3. Instructions désormais non supportées 194
4. Extension de fichier, noms de fichier 194
5. Actions par défaut 195
6. Axes digitales vc axes analogiques. 195
7. Instructions digitales Type 1 196
8. Manette des gaz non présente 196
9. Macros – caractères interdits 196
10. RPT 196
11. le caractère de commentaire // 197

1. Ce que nous avons pour vous !

1.1 Introduction

Voici la dernière génération des contrôleurs de “haute définition” Trustmaster : Le Hotas Cougar.
Le Pack contient un contrôleur moulé dans une carapace en acier, un CD-ROM contenant un nombre important d’utilitaires et de programmes, le guide de référence, et le traditionnel guide d’installation rapide.

Parmi les logiciels fournis sur le CD-ROM, vous trouverez tous les « softs » nécessaires à la mise en route et à l’utilisation du « Hotas Cougar » : le panneau de configuration du Hotas (décrit plus bas dans ce chapitre), ainsi que deux logiciels « Foxy » (incluant des éléments tel que le « compiler » et les utilitaires « Korgy ») et « Foxy Gui ».

Nous n’aborderons pas ici le panneau de configuration du Hotas, pour plus d’informations concernant celui-ci continue la lecture de ce manuel.
Foxy et Foxy GUI, cependant, ne sont pas approfondis dans ce manuel. Pour toutes explications concernant l’utilisation de ces logiciels, veuillez vous connecter sur leurs sites internet respectifs.
Foxy est la clé pour programmer facilement chaque fonction de votre joystick, puissant et complexe utilisant des macros, il est néanmoins d’une facilité déconcertante.
Le programme Foxy GUI vous permet d’affecter précisément, à l’aide de quelque clic de souris, une gamme de combinaison de touches au Hotas Cougar, ainsi vous êtes capable de configurer votre joystick sans connaissances particulières du code de programmation Trustmaster.

Voilà, maintenant mettez en route votre Hi-fi avec votre CD favori, servez vous un verre et préparez vous à passer les prochains jours (et nuit) dans les profondeurs de votre manuel de référence !

1.2 Configuration de votre joystick

Veuillez vous référer à votre manuel d’installation rapide.

1.3 Premier contact avec votre Guide de référence :

Manifestement, un contrôleur d'avant-garde comme le Hotas Cougar exige un support logiciel conséquent afin de pouvoir exprimer tout son potentiel. Lorsque vous aurez installé votre Cougar, les logiciels fournis sur le CD et que vous aurez vérifié dans le panneau de configuration du Hotas (CCP)et également dans celui de Windows que tout semble fonctionner, vous vous demanderez sans doute par où commencer. Heureusement vous avez également installé Foxy, et vous l'avez certainement démarré (ceci après le CCP – c'est important). Ceci dit nous allons commencer par les bases et vous apprendre ce que peut faire votre Hotas Cougar.

Niveau 1 : Utilisation de Base

Vous aurez le plaisir d'apprendre qu'à ce stade vous n'avez rien d'autre à faire pour utiliser votre joystick. Néanmoins amener tous les axes au maximum de leur course en maintenant cette position quelques secondes ceci afin de calibrer automatiquement le Hotas. Maintenant quittez tous les programmes du Hotas, démarrez vos jeux, et vous pouvez ainsi utiliser votre Cougar. Si le jeu vous le permet configurez directement les boutons du Hotas. Il va reconnaître les axes et les boutons du système et avec un peu de chance il va leur assigner des fonctions de vol normales il va même reconnaître les axes particuliers comme l'échelle et les réglages d'antenne et les affecter sur la manette des gaz. Dans ce cas le Cougar est en mode Direct X, ce qui veut dire que les boutons et les axes ne sont pas programmés et qu'ils peuvent être directement configurés à l'aide du jeux.

Niveau 2 : Programmation du Cougar à l'aide des fichiers fournis pour vos jeux.

L'étape suivante va vous permettre de programmer votre Cougar à l'aide de fichiers développés pour vos différents jeux, une trentaine de jeux ont été étudiés. Ces fichiers de réglages se trouvent sur le CD et peuvent être utilisés à l'aide du panneau de contrôle du Cougar (CCP). Il existe un moyen plus simple de configurer le Cougar : utiliser les programmes Foxy et Foxy GUI.

FoxyGUI : Démarrer FoxyGUI, Suivez les instructions à l'écran, choisissez le jeux désiré, appuyez sur la touche « download », quittez FoxyGUI, démarrer votre jeux.

Foxy : Démarrer Foxy, allez dans le menu « Editor's Favourites », sélectionnez le jeux désiré. Deux fichiers s'ouvrent, celui de gauche affiche le joystick et celui de droite affiche le fichier macro.
Pour l'instant vous n'avez besoin de rien d'autre. Maintenant aller dans le menu « Download », cliquez sur « download ». Cela va alors programmer votre Cougar à l'aide des fichiers que vous venez d'ouvrir. Voilà, votre Cougar est configuré pour votre simulateur. Vous pouvez également cliquer sur l'image et/ou sur le texte afin d'afficher ce que les concepteurs des ces fichiers ont à dire concernant ceux-ci et la façon de les utiliser avec votre jeux.

Niveau 3 : Apprenez à programmer votre Cougar.

Nous avons développé plusieurs façons d'apprendre à programmer votre Cougar, vous pouvez choisir ci-dessous la manière qui vous convient le mieux.

1. Le chapitre suivant I, *(2.1 – Comprendre les bases de la programmation Trustmaster)* comme le fichier d'aide de Foxy, donne une bonne idée des bases à acquérir pour programmer le Cougar, cela de façon très simple et conviviale.

2. Dans le menu Foxy, essayez les macros associées au joystick. Beaucoup de personnes apprennent à programmer leur contrôleurs TM à l'aide ces deux façons.

3. Il existe également dans ce même menu, des fichiers « Tutorial files »que vous pouvez activer, lesquels ouvrent des fichiers d'aide et d'apprentissage dans « Foxy ». Vous pouvez charger, modifier, et constater les effets de vos changements. C'est une bonne manière d'appréhender les bases de la programmation.

4. De la même manière FoxyGUI fourni une façon assez simple de programmer votre Cougar et aussi bien expliqué qu'avec le logiciel Foxy. Malgré tout vous préférerez sans doute Foxy parce qu'il est plus rapide et plus puissant lorsque vous avez compris les bases de son utilisation.

5. A tout moment lorsque vous utilisez Foxy, vous pouvez obtenir de l'aide en appuyant sur F1. Vous pouvez également, en sélectionnant un mot et en pressant la touche F1, obtenir l'aide directe. Le fichier d'aide est très important et il contient de nombreux détails et explications comprises dans ce manuel.

6. Utilisez « Composer et Korgy » du menu Insert de Foxy… il vous sera rarement nécessaire de vous plonger dans les profondeurs du manuel concernant ces deux composantes de Foxy.

Une chose encore, ne soyez pas stresser, programmer le Cougar est une chose vraiment très facile. Prenez juste le temps de lire l'aide nécessaire et vous

récolterez les fruits de votre patience. Une fois que vous aurez bien approchez les bases de la programmation et tous ses avantages vous aurez du mal à retourner vers une « approche graphique » de la configuration, très commune avec les logiciels de configuration de certains contrôleurs

2. LES PRINCIPES FONDAMENTAUX

2.1 Les principes de base de la programmation Thrustmaster

2.1.1 Introduction

En parlant de programmation, les joysticks et manettes de gaz Thrustmaster ont souvent été pris pour la référence que l'on a utilisé pour les comparer aux autres systèmes. Malheureusement, ils avaient également la réputation d'être difficile à programmer, cela est probablement dû au fait que les programmes fournis avec ceux-ci étaient basés sur le DOS, et également parce que les gens ne prenaient pas le temps nécessaire pour comprendre ces contrôleurs. Nous pouvons vous assurer que la programmation du système vous semblera plus facile que l'apprentissage de certains simulateurs de vol.

Ce que nous proposons de faire ici est de partir des bases en supposant que vous n'avez aucune expérience dans la programmation des systèmes Thrusmaster. Il est dommage d'employer le terme « programmation », un terme normalement associé au développement de logiciels et aux langages informatiques. Nous allons seulement développer des fichiers qui vont permettre d'assigner des commandes à chacun des boutons du joystick et de la manette des gaz.

2.1.2 LE concept du HOTAS

Maintenant dans beaucoup de simulateur, à l'aide du clavier, vous pouvez voler en contrôlant les armes, les commandes à l'intérieur du cockpit etc. On peut rendre ceci un peu plus réaliste en ajoutant un joystick, une manette des gaz et des palonniers, plus généralement appelé « Contrôleur de vol ». Malgré tout vous avez encore besoin d'utiliser votre clavier. Mais que ce passe t il si le joystick et la manette des gaz sont équipés d'assez de boutons permettant d'avoir le même

effet que les touches du clavier ? Dans ce cas vous pouvez garder vos mains sur le joystick et la manette des gaz et ainsi vous concentrer sur votre vol et votre mission. C'est le principe du HOTAS, il vous permet de garder vos mains sur le système de vol. (En anglais : **H**ands **O**n **T**hrottle **A**nd **S**tick at all times)

2.1.3 Comment avons nous conçu le HOTAS pour les simulateurs et les jeux ?

Assez simplement, nous avons programmé les contrôleurs afin qu'ils imitent un clavier et les touches que vous utilisez pour vos simulateur de vol. Ceci en utilisant deux fichiers, le fichier « **joystick file »** lequel détermine quels sont les boutons et « hat » que vous voulez utiliser pour y affecter les commandes du clavier. De plus le fichier « macro file », contient des *"macros"* qui indiquent quelles action ont les commandes du clavier dans vos simulateur de vol. Commençons par prendre contact avec ces deux fichiers .2.1.3 Comment avons nous conçu le HOTAS pour les simulateurs et les jeux?

2.1.4 Le fichier "joystick file" – Principes de programmation

Nous avons vu plus haut que le fichier « joystick File" était utilisé pour affecter aux boutons et "hat" du joystick les différentes commandes du clavier. Il semble impropre de parler uniquement de fichier « joystick ». En effet ce fichier va vous permettre de configurer tout les boutons de tous vos contrôleurs (manette des gaz et palonnier).

Tous les boutons de votre joystick et manette des gaz ont un nom particulier, ceci afin de pouvoir les configurer sans ambiguïté. Nous n'allons tous les énumérer ici. En fait, vous n'aurez pas besoin de tous les connaître, le programme « Foxy ».vous permettra de les apprendre tout au long de son utilisation. Nous allons seulement faire connaissance avec certains d'entre eux, afin de pouvoir commencer l'apprentissage de leur programmation.

Disons que nous voulons programmer le bouton S2 du joystick afin qu'il active le pilote automatique. Vous pouvez voir sur la figure ci-dessus la position du bouton S2. Il est situé sur la face avant du joystick sur la gauche du « Hat » blanc, appelé Hat1. Revenons au bouton S2, supposons que le pilote automatique sois commandé par la touche A du clavier (comme c'est le cas dans beaucoup de simulateur).

Nous avons donc besoin de dire au joystick de « produire » un A à chaque fois que l'on appuie sur le bouton S2. Dans le fichier « joystick file », qui est un fichier texte, un bouton est identifier par le terme "BTN", dans ce cas nous voulons utiliser le bouton S2, nous aurons alors l'expression BTN S2, de plus nous voulons lui affecter la touche A du clavier, l'expression devient alors :

BTN S2 a

Facile, non ? Programmons maintenant le mouvement vers le haut du Hat1, afin qu'il corresponde à la touche F1 du clavier. On suppose que dans votre simulateur cela corresponde à la « vue avant ». Tous les Hat sont en réalité des boutons, donc comme précédemment, nous allons utiliser l'état bouton (BTN) associer au HAT 1 UP (vers le haut) : on a alors H1U et on associe la touche F1 on obtient donc :

BTN H1U F1

Ces expressions sont la base de la programmation Trustmaster.

De nos jours, dans certains simulateurs il n'est pas rare d'avoir plus d'une centaine de commandes. Essayer de retenir toutes ces commandes peut devenir un cauchemar. Nous allons donc pouvoir ajouter une remarque (REM) qui nous rappellera l'affectation de la commande.
Exemple :

BTN S2 a REM Activer / Désactiver le pilote automatique
BTN H1U F1 REM Vue Avant

Mais il existe une façon beaucoup plus simple de procéder. Vous pouvez utiliser les macros et le fichier macro. Avant de parler de ceci, il est utile de noter que la déclaration REM peut être placée n'importe où dans le fichier et que toutes les expressions ou caractères se trouvant après REM sont ignorés par le joystick. On utilise souvent cette manière de faire pour placer un titre ou un commentaire au début d'un fichier

2.1.5 Macros et « macro file »– Principes de programmation

Avant de développer le fichier macro, expliquons ce qu'est une macro. A la page précédente nous avons parlé du fichier joystick et nous avons travaillé avec deux « commandes » :

BTN S2 a   REM Activer / Désactiver le pilote automatique
BTN H1U F1   REM Vue Avant

Une macro est un mot que nous utilisons pour nous rappeler facilement quelle est la fonction d'une touche de clavier ou d'un groupe de touches dans notre simulateur de vol.
Exemple :

Autopilot = a
Forward_view = F1

Nous pouvons alors transformer les commandes de notre fichier joystick ainsi :

BTN S2 Autopilot
BTN H1U Forward_view

Il n'est peut-être pas évident de penser que cette méthode est plus simple mais lorsque vous avez un fichier joystick qui contient plus d'une centaine de commandes, cela s'impose.

Où allons nous insérer ces commandes macro ?

Elles se trouvent dans leur propre fichier, le fichier « macro file ».En conclusion, on peut dire que ce fichier contient toutes les macros décrivant les commandes claviers de votre simulateur et leurs actions, et le fichier « joystick file » assigne ces commandes aux boutons de votre joystick et de votre manette des gaz. Vous devriez, par conséquent pour un jeu ou un simulateur, avoir les fichiers Thrustmaster par paires (joystick et macro file).C'est la raison pour laquelle, dans le programme Foxy, l'écran principal contenant l'éditeur est divisé en deux, une partie montre le fichier « Joystick file » et l'autre le fichier « Macro file ». Nous allons dire pour notre exemple qu'il concerne le simulateur de vol Falcon 4, nous aurons donc deux fichiers à sauvegarder, le fichier « joystick file » en Falcon 4.tmj et le fichier « macro file » en Falcon 4.tmm (tmj = **TM J**oystick File, tmm = **TM M**acrofile.)

2.1.6 Comment le fichier « joystick file » sait-il quel fichier « macro file » contient ses macros ?

Nous avons vu plus haut les bases de la conception des fichiers joystick et macro, et normalement cela doit vous paraître simple…
Imaginons que vous avez une trentaine de simulateurs de vol, cela veut dire que vous avez une trentaine de fichiers joystick et une trentaine de fichiers macro dans votre répertoire « Foxy's Files ».

Pour l'instant les fichiers ne sont pas liés. Pour compléter notre fichier exemple nous devons donc indiquer au fichier joystick quel fichier macro contient les commandes macro appropriées (Macro DEFinitions MDEF). Nous allons le faire en indiquant dans le fichier joystick au moyen de l'expression USE MDEF :

USE MDEF Falcon 4.tmm

Le programme Foxy va lire la ligne USE MDEF dans le fichier joystick lorsqu'il va ouvrir le fichier, trouver le fichier macro correspondant dans l'exemple : le fichier Falcon 4.tmm, et ainsi utiliser les macros de celui-ci pour programmer le fichier joystick.

2.1.7 Résumons ce que nous avons appris plus haut…

Regardons ensemble ce que cela donne dans le fichier « joystick file » et le fichier « macro files ».

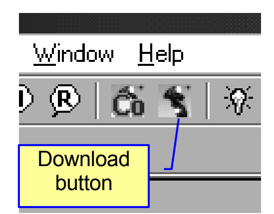
Joystick file *(Falcon 4.tmj)*	**Macro file** *(Falcon 4.tmm)*
REM -- REM Falcon 4.tmj REM Falcon 4 joystick file REM REM Rem statements don't do anything. REM We use them to add comments REM -- REM We tell the joystick file which REM macro file contains its macros USE MDEF Falcon 4.tmm REM Now we program some buttons by REM assigning macros onto them REM from the macro file BTN S2 Autopilot BTN H1U Forward_view	REM -- REM Falcon 4.tmm REM REM Falcon 4 macro file REM -- REM Macros make it much easier for REM us to remember what actions REM keyboard keys perform in our REM flight sim. REM Macro definitions start here Autopilot = a Forward_view = F1

Oui, ces commandes ne sont pas vraiment les commandes de Falcon 4 – Nous avons juste voulu illustrer les propos des chapitres précédent
Nous avons donc vu :

1. Le fichier macro contient des macros qui décrivent l'action des commandes clavier dans votre simulateur.
2. Le fichier joystick assigne ces macros, aux boutons de votre joystick et de votre manette, via la commande « BTN ».
3. Le fichier joystick obtient les macros du fichier macro approprié via la commande « USE MDEF ».
4. La commande « REM » est un moyen simple d'obtenir des commentaires à propos du fichier.
5. Les fichiers joystick et macro files sont des fichiers texte simples, avec respectivement les extension .tmj et .tmm , ils sont situés dans le même répertoire sur votre disque dur. Par défaut ce répertoire est le répertoire « Foxy's Files ».

2.1.8 Charger le fichier joystick sur le système de vol

Bien! Nous avons parcouru les bases de la conception des fichiers essentiels pour votre système de vol. Il nous reste à savoir comment charger ces fichiers sur les contrôleurs ?



La façon la plus simple est d'appuyer sur le bouton « Download » sur la barre de tache du programme Foxy, ou encore vous pouvez appuyer sur la touche « F12 » de votre clavier. Après quelques secondes, le fichier est transféré, ou plus justement il est chargé sur votre système Trustmaster. A ce stade, vous êtes prêt à voler avec un joystick et ses boutons programmés comme indiqué dans le fichier « Joystick file ». Simple ! Avant d'aller plus loin, il nous allons voir ce qu'il se passe lorsque vous chargé un fichier « joystick file » sur votre système.

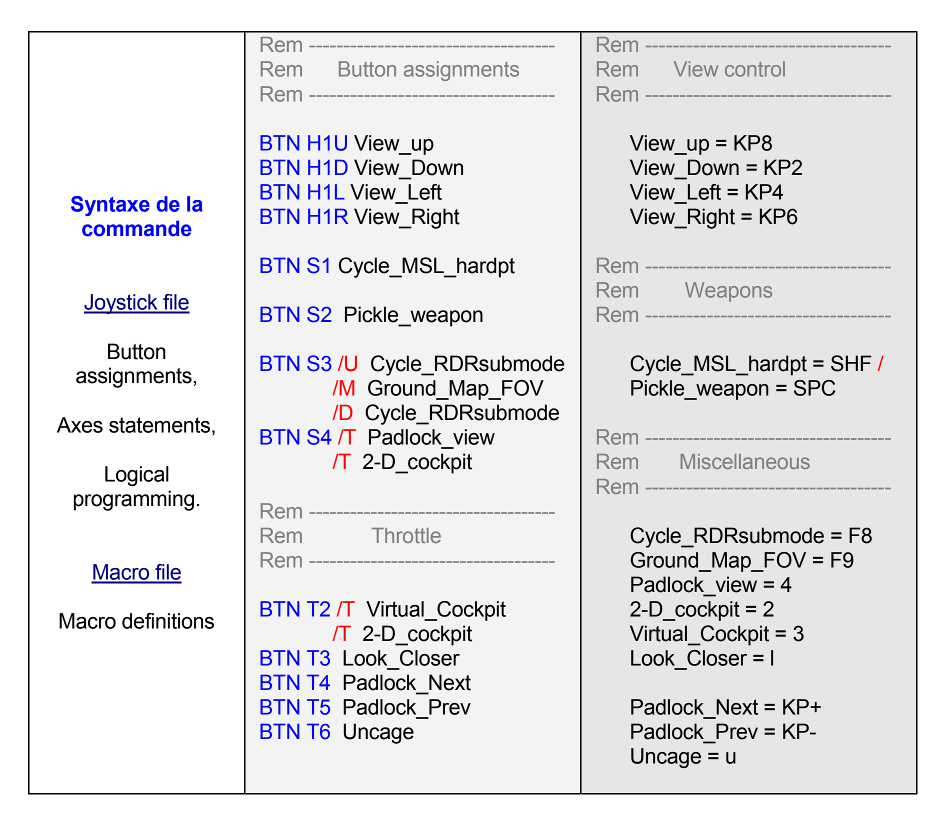
Le fichier « joystick file » est envoyé à une des applications du système Trustmaster Cougar, appelée le « Compiler ». Il a pour but de convertir le texte contenu dans le fichier « joystick File » et les instructions contenues dans le fichier « macro file » en un langage compris par le joystick et la manette des gaz. Lorsque le fichier compilé ne contient pas d'erreur, il est chargé dans le système. Il envoie alors un message au programme Foxy pour lui indiquer que la compilation c'est bien passée. Dans ce cas le joystick et la manette

2.1.9 Structure des fichiers « joystick et macro files »

Avant d'aller plus, nous allons nous attarder sur les règles générales concernant la structure des fichiers « joystick et macro files ». L'exemple ci-dessous nous aidera à avoir une idée sur la question. A ce stade ne vous inquiétez pas si vous ne comprenez pas toutes les lignes de commande. Nous voulons juste vous donner une idée de ce que contiennent les fichiers joystick et macro, ainsi que leur mise en forme. Il est aussi à noter que le fichier joystick contient une section appelée « configuration statements », que nous développerons plus tard.

Sections	**Joystick file** *(Falcon 4.tmj)*	**Macro file** *(Falcon 4.tmm)*
Title Not compulsory, but a good idea.	Rem ----------------------------------- Rem Falcon 4.tmj Rem Rem Falcon 4 joystick file Rem Rem last modified 1st Jan 01 Rem Rem -----------------------------------	Rem ----------------------------------- Rem Falcon 4.tmm Rem Rem Falcon 4 macro file Rem Rem last modified 1st Jan 01 Rem Rem -----------------------------------
Configuration statements *(**only** in the joystick file)*	Rem Rem Configuration statements Rem USE MDEF Falcon 4 USE RATE (60) USE TG1 AS DX1 USE S2 AS DX2	Rem Rem Configuration statements Rem Rem don't go into macro files Rem Rem So macro definitions start here

(voir page suivante!)



Syntaxe de la commande Joystick file Button assignments, Axes statements, Logical programming. Macro file Macro definitions	Rem ------------------------------------ Rem Button assignments Rem ------------------------------------ BTN H1U View_up BTN H1D View_Down BTN H1L View_Left BTN H1R View_Right BTN S1 Cycle_MSL_hardpt BTN S2 Pickle_weapon BTN S3 /U Cycle_RDRsubmode /M Ground_Map_FOV /D Cycle_RDRsubmode BTN S4 /T Padlock_view /T 2-D_cockpit Rem ------------------------------------ Rem Throttle Rem ------------------------------------ BTN T2 /T Virtual_Cockpit /T 2-D_cockpit BTN T3 Look_Closer BTN T4 Padlock_Next BTN T5 Padlock_Prev BTN T6 Uncage	Rem ------------------------------------ Rem View control Rem ------------------------------------ View_up = KP8 View_Down = KP2 View_Left = KP4 View_Right = KP6 Rem ------------------------------------ Rem Weapons Rem ------------------------------------ Cycle_MSL_hardpt = SHF / Pickle_weapon = SPC Rem ------------------------------------ Rem Miscellaneous Rem ------------------------------------ Cycle_RDRsubmode = F8 Ground_Map_FOV = F9 Padlock_view = 4 2-D_cockpit = 2 Virtual_Cockpit = 3 Look_Closer = l Padlock_Next = KP+ Padlock_Prev = KP- Uncage = u

Pour l'instant c'était une introduction rapide aux bases de la programmation. Maintenant il est temps de nous intéresser en détail à la structure du fichier « joystick file ». Nous allons débuter par une section que nous avons déjà évoqué plus haut : la déclaration des boutons et « hat », ainsi qu'à leur appellation. Nous allons également décrire plus en profondeur les macros.

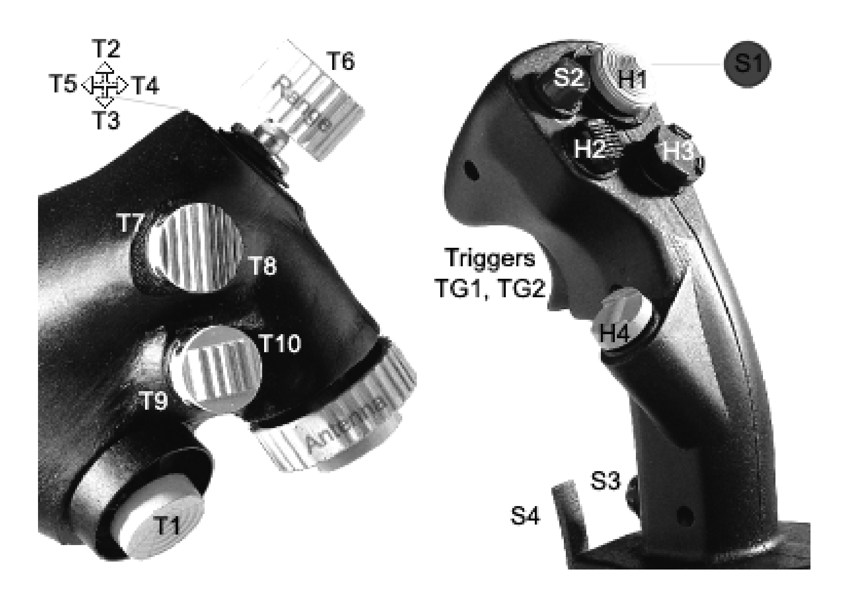
3. Déclaration des boutons et Macros

3.1 Déclaration des boutons et syntaxe du langage TM

Le Hotas cougar est constitué de plusieurs axes, de boutons, de « hat », de gâchette etc. Tout ce qui n'est pas un axe peu être programmé en déclarant un bouton, en utilisant la syntaxe ci-dessous :

Syntaxe de la commande
BTN Button_name KeySequence and/or macro/s

Dans ce cas :
Button_name indique le bouton qui doit être programmé :

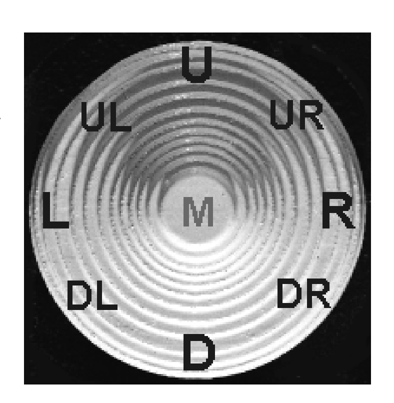


La manette des gaz a :	10 boutons :	de T1 à T10
Le joystick a :	4 hats(chapeaux) :	de H1 à H4
	4 switches(interrupteurs) :	de S1 à S4
	2 stage trigger (gâchettes):	TG1 TG2

Exemples :

BTN T3 y Rem "Roger, understood old fruit"
BTN S2 Eject
BTN T4 Chaff Flare Rem Time for bowel movements
BTN S4 h e l l o Rem Notice the spaces – it's not a macro

Chaque « hat » a 9 positions programmables :



En général on utilise seulement les 4 positions principales. Pour le « Hat »1, on peut avoir l'exemple suivant

BTN H1U Look_up
BTN H1R Look_right
BTN H1D Look_down
BTN H1L Look_left

Les positions intermédiaires peuvent aussi être programmées :
BTN H1UL View_UL

Ainsi que la position centrale :

BTN H1M View_forward

NOTES

1. *La déclaration d'un bouton n'a pas besoin de se située exactement au début d'une ligne. Mais on ne doit avoir qu'une déclaration par ligne et on ne peut déclarer un bouton qu'une seule fois dans un fichier. Donc si vous avez :*

BTN S2 a b c
BTN S3 d e f

Alors ces déclarations sont bonnes, mais si vous avez plus loin dans le fichier :

BTN S3 g h I

Alors le « compilateur » va générer un message d'erreur vous informant que vous avez un doublon dans votre fichier « duplicate button statement ».

2. *IL ne doit y avoir qu'un seul espace entre la commande BTN et le nom du bouton, donc :*

BTNS2

Va générer une erreur.
Vous devez également avoir un espace après le nom du bouton, donc :

BTN S2a b c

Va également générer une erreur de compilation.

3. *Un caractère ou une commande macro ne sera généré qu'une seule fois lorsque vous appuyer sur le bouton concerné (même si vous maintenez ce bouton enfoncé). Si vous voulez une répétion de la commande ou maintenir le bouton en position enfoncé alors il vous faudra insérer dans la ligne un attribut, par exemple* /A *auto-repeat (répétion) ou* /H *hold (maintenir).La barre d'attribut (/) sera décrite plus loin dans le guide de référence..*

NOTES

Nous allons nous écarter légèrement du sujet et parler de la position centrale du « Hat » .Nous ne pensons pas que vous allez essayer ceci pour l'instant mais plus tard vous pourrez utiliser cette remarque pour l'insérer dans un de vos fichiers. Le centre de chaque « Hat » peut être programmé en ajoutant « M », comme dans l'exemple ci-dessous .Il faut noter que si les commandes /P, /R *(voir plus loin les remarques concernant la barre d'attribut) sont programmées sur l'une des positions du « ha », l'option* /R sera générée en même temps que la commande « *M* ». *Donc :*

```
BTN H1U     /P 1
            /R 2
BTN H1M a
```

Va générer, lorsque l'on appuiera sur le « hat 1 :
"1", puis "a" et "2" de façon simultanée.

Si vous voulez être sûre que la commande H1M soit exécutée après la commande H1U /R, *alors il vous faudra insérer un delai (voir plus loin dans les notes) à la commande* H1M *comme indiqué ci-dessous :*

```
BTN H1M DLY(60) a
```

3.2 Syntaxe des commandes clavier Thrustmaster

Revenons à la syntaxe des commandes :

> Syntaxe de la commande
>
> BTN Button_name KeySequence and/or macro/s

Regardons de plus prêt la partie – « key sequence ».

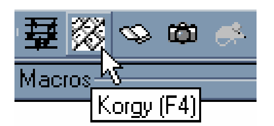
Si nous voulons assigner une touche du clavier, soit dans une macro ou sur le joystick, alors nous aurons besoin de connaître la style particulier qu'a « Thrustmaster » pour identifier chaque touche du clavier. Par exemple, appuyer sur la touche 5 du clavier principal à peut-être un résultat complètement différent dans votre simulateur que la frappe de la touche 5 du pavé numérique. Nous devons donc être capable de distinguer la touche 5 et la touche KP5 (pavé numérique).C'est pourquoi il existe une syntaxe Thrustmaster pour les commandes clavier :

ESC	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	
`	1	2	3	4	5	6	7	8	9	0	-	=	BSP
TAB	q	w	e	r	t	y	u	i	o	p	[	]	\
CAPS	a	s	d	f	g	h	j	k	l	;	'	ENT	
LSHF	z	x	c	v	b	n	m	,	.	/	RSHF		
LCTL	LALT	SPC	RALT	RCTL									

PRNTSCRN	SCRLCK	BRK

INS	HOME	PGUP
DEL	END	PGDN
	UARROW	
LARROW	DARROW	RARROW

NUML	KP/	KP*	KP-
KP7	KP8	KP9	KP+
KP4	KP5	KP6	
KP1	KP2	KP3	KPENT
KP0		KP.	



La façon la plus simple de nous assurer que nous utilisons la bonne syntaxe est d'utiliser « Korgy », le clavier virtuel dans le programme Foxy. Vous pouvez aussi accéder à ces informations à l'aide du menu aide, en sélectionnant "Keyboard Syntax" dans le menu menu principal.

NOTES

1. LA syntaxe pour les touches liées *(lorsque vous utilisez les touches Shift, ALT ou CTRL) est* SHF a, ALT b, CTL c. Ce n'est pas la même chose que d'utiliser LSHF a, LALT b or LCTL c. Par exemple*,* LSHF à pour signification, appuyer sur la touche shift située à gauche sur le clavier*, puis la relâcher, et appuyer sur le "a", et la relâcher. Le « compiler » utilise par défaut les touches Shift, ALT, CTRL situées à gauche sur le clavier pour les touches liées.*

2. Il existe des touches réservées : ( ) { } < > et elles sont prog*rammées à l'aide de la commande* SHF :

(= SHF 9
) = SHF 0
{ = SHF [
} = SHF]
< = SHF ,
> = SHF .

3. Il est bon de prendre l'habitude d'utiliser SHF, ALT *or* CTL en relation avec une autre touche*, ainsi lorsque l'on désire une lettre majuscule :*

BTN S1 SHF p
BTN S1 P

Il est préférable de prendre cette bonne habitude, les deux commandes sont correctes, elles donnent toutes les deux un P.

4. La syntaxe est basée sur un clavier Américain. *Il vous sera sans doute nécessaire de reproduire une touche qui n'existe que sur un autre type de clavier. Dans ce cas, vous pourrez le faire en utilisant directement les codes USB … nous parlerons de ça un peu plus loin dans le guide de référence. Il faut noter que cela risque d'arriver de façon très rare, mais on ne sait jamais !*

5. La syntaxe a changé par rapport au *TM F22 .Par exemple, le suffixe AUX et d'autre « clés » a été supprimées.*

3.3 Les Macros et les commandes macro

Dans l'introduction, nous avons vu le concept des macros et du fichier macro. Nous avons pris deux exemples :

Autopilot = a
Forward_view = F1

Maintenant que vous connaissez la syntaxe des touches du clavier, nous pouvons aborder la création de fichiers macro et des commandes macro. Foxy possède plusieurs utilitaires qui vont vous permettre de créer des macros en utilisant une syntaxe correcte et les règles appropriées. Ces utilitaires sont : « Macro Wizard, Speedy et Korgy ». Utilisez la documentation de *Foxy pour plus d'informations*. Avant d'approfondir le sujet, nous allons juste ajouter une petite remarque. En effet, il va sans doute vous paraître ennuyeux et pénible de reprendre toutes les commandes de votre simulateur et de créer les macros associées à ces commandes. Si le jeu possède un fichier d'aide, vous pourrez certainement récupérer toutes les commandes clavier et ainsi les transformer sous forme de macro, en respectant les règles décrites ci-dessous. Il faut souligner un dernier point, nous avons souvent reçu des fichiers de personnes qui ont eu des problèmes malgré le bon chargement des commandes sur les contrôleurs, parce que le « compiler » générait des erreurs qui n'étaient immédiatement observées. Nous avons appris par expérience d'après ces fichiers, qu'il faut d'abord passer le fichier « macro file » en revue
Afin de trouver une syntaxe incorrecte, ou des macros qui ne respecterai pas les règles ci-dessous. On peut dire que dans 90% des erreurs de compilation sont dues à des erreurs contenues dans le fichier « macor file ». Prenez le temps de créer vos propres fichiers macro.

Maintenant passons aux règles et normes …

1. Le titre des macros ne peut contenir d'espace. Utilisez _ « underscore » _ ou – « moins » :

My macro is = b

N'est pas une commande permise, préférez :

My_macro_is = b
My-macro-is = b

2. Assurez vous de la présence d'un espace avant et après le signe "="situé après le titre de la macro. Donc on peut dire que ces deux macros :

Autopilot= a
Autopilot =a *sont incorrectes.*

3. On ne peut pas utiliser les caractères suivant dans le titre des macros :

= < > { } ( ) ^ , e*space*

4. Essayez de ne pas utiliser les majuscules pour le titre des macros, (Exemple : RADAR_RANGE_INCREASE). Vous pouvez le faire, mais la lecture est plus facile lorsque vous ne le faites pas. Il est bon de réserver cette syntaxe pour

les commandes TM (Exemple : MDEF), ou les abréviations (Exemple : HUD) .

5. Le titre des macros ne fait pas la différence entre les deux syntaxes ci-dessous, donc :

MyMacro = a
mymacro = a ont la même signification.

NOTES

Rentrons maintenant dans le vif du sujet !

Les macros peuvent être imbriquées les une dans les autres : Vous pouvez utiliser une macro pour appeler une autre macro :

Macro_1 = a b c
Macro_2 = Macro_1 d e f

Vous pouvez avoir jusqu'à *20 macros à imbriquées dans une macro :*

Macro-1 = a Macro-2
Macro-2 = b Macro-3
Macro-3 = c Macro-4
... etc...
Macro-20 = d

Mais pas plus de 20 Si cela devait arriver, nous aurions :.

Macro-1 = a Macro-1

Le « compiler » va alors générer une erreur exprimée comme ceci : "Macro looping too long; two macros might be calling themselves."

3.4 Modifier une déclaration

Nous allons nous intéresser à quelques déclarations simples, comme ci-dessous :

BTN T4 a

Cette déclaration va produire un seul "a" qui aura été généré lorsque l'on aura appuyé et relâcher le bouton T4 situé sur la manette des gaz. Vous allez peut-être avoir besoin de générer plusieurs « a ».Sur un clavier vous pouvez appuyer sur une suite de touches afin de générer une suite de caractères, avec des délais entre chaque touche ou encore tout une suite d'action ou de répétition, etc... **Le**

but d'un bon système de vol, est de vous permettre de reproduire n'importe quelles actions que vous auriez exécutées à l'aide d'un clavier.

C'est là que nous allons introduire la notion d'attribut de déclaration. Nous voulons être capable de produire plus d'un seul caractère à partir d'un bouton.

L'attribut de déclaration est une commande qui est utilisée pour changer le comportement d'un caractère programmé sur un bouton. They come in 5 flavours:

1. Les attributs /U, /M, /D, /I, /O, /P, /R, /T, /A, /H

```
BTN S4 /H b   Rem Wheelbrakes
BTN T3 /A c f  Rem Chaff and flares
```

2. Déclaration de délai ou de répétition DLY( ), RPT( )

```
BTN T6 1 DLY(60) 1 DLY (60) 2 Rem Request Vector For Recovery TAW
BTN S2 RPT(6) c  Rem 6 chaffs please … like right now would be good!
```

3. Regroupement de caractères en utilisant ( ), { }, < >

```
BTN T2 (a b c)
BTN T3 {a b c}
BTN T4 /P <a b c>
       /R d
```

4. Utilisation des boutons DirectX (Direct Input) DX

```
USE TG1 AS DX1
BTN H2U DX1
USE ALL_DIRECTX_BUTTONS
```

5. Utilisation des touches « Down, Up » et des codes USB KD( ), KU( ), USB( )

```
BTN H4U KD(a) DLY(60) KU(a)
BTN H4D /P USB (D51) /R USB (U51) Rem 'Down arrow'
```

Commençons par évoquer la barre d'attribut…

3.5 Les attributs

Il existe en tout 10 attributs, et ils peuvent être regroupés en trois catégories basées sur l'action qu'ils provoquent sur les boutons et leurs déclarations :

Augmentation du nombre de positions programmables :

/U, /M, /D utilisent le « switch Dogfight » de la manette des gaz (T7, T8)
/I, /O utilisent le bouton S3 du joystick

Séparation des macros sur un bouton :

/T interrupteur à bascule (on ou off)
/P, /R appuyer et relâcher

Répétition de caractères ou non :

/A répétition automatique
/H Maintenir

Règles de la barre d'attribut et hiérarchie :

En fait, vous devez comprendre où vous pouvez utiliser un attribut et dans quel ordre.

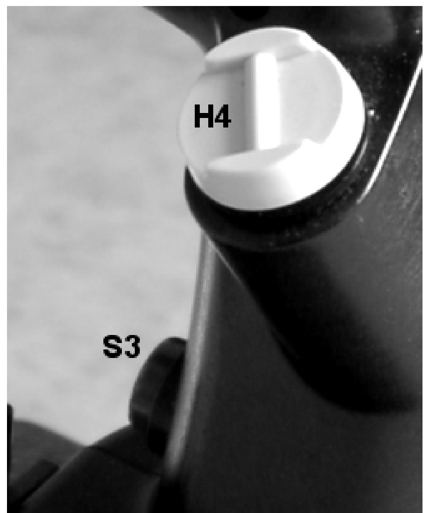
3.5.1 Augmentation du nombre de positions programmables :

Le « switch Dogfight » de la manette des gaz, peut être actionné dans 3 positions différentes – haut « Up » (/U), milieu « Middle » (/M), bas « Down » (/D). De la même façon, le bouton S3 peut être actionné en position (/I) ou (/O), voir le tableau plus bas. Vous pouvez utiliser une combinaison de tout ceci afin d'augmenter le nombre de positions programmables d'un bouton/axe/ou « hat ».

Attributs de déclaration

/U, /M, /D using the Throttle's Dogfight switch (T7, T8)
/I, /O using the Joystick's Button S3 switch





3.5.1.1 /U, /M, /D - Up, Middle, Down

Vous pouvez augmenter le nombre de positions programmables sur un bouton, un axe,ou un hat en utilisant la position du switch « dogfight » associé aux barres d'attribut /U, /M et /D. Par exemple :

```
BTN H4U /U a
        /M b
        /D c
```

Lorsque l'on appuie sur la HAT4, cela donne :

- un "a" si le switch dogfight est en position haute (up) (/U)
- un "b" si le switch dogfight est en position moyenne (middle) (/M)
- un "c" si le switch dogfight est en position basse (down) (/D)

Vous devez placer les barres d'attribut /U, /M, /D sur des lignes différentes. Si ce n'est pas le cas alors cela va générer une erreur de compilation. De plus l'ordre indiqué ci-dessus devra être respecté, donc :

```
BTN H4U   /D a
          /M b
          /U c
```

Va générer une erreur de compilation.

3.5.1.2 /I, /O - In, Out

Vous pouvez augmenter le nombre de positions programmables d'un bouton, d'un hat ou d'un axe en utilisant le bouton S3, avec les options /I et /O. Par exemple :

```
BTN H4D   /I 1
          /O 2
```

Lorsque l'on appuie sur le hat 4, cela donne :

- un "1" si l'on appuie sur le bouton S3 (/I = **I**n)
- un "2" si le bouton S3 n'est pas utilisé (/O = **O**ut)

Combinaison de /U, /M, /D avec /I, /O :

Vous pouvez également combiner ces barres d'attribut. Par exemple :

```
BTN H4R   /U /I   Engage_my_target
             /O   Break_right
          /M /I   Camera_right
             /O   Next_waypoint
          /D /I   Engine_right
             /O   View_right
```

Alors lorsque l'on bascule le hat4 sur la droite, on peut avoir 6 fonctions différentes, cela va dépendre de la position du Dogfight switch (/U, /M, /D) et du bouton S3 (/I, /O).

NOTES

1. Les barres d'attribut /U, /M, /D*, doivent précéder les options* /I, /O*.*

2. Vous ne pouvez pas utiliser /U, /M, /D avec les « *dogfight switch »de la manette des gaz (boutons T7 et T8)*

3. Vous devez placer /I, /O sur des lignes distinctes.

4. Vous devez placer l'option /I avant /O*. Donc :*

```
BTN H4D   /I 1
          /O 2
```

Est une déclaration valide mais les deux ci-dessous :

```
BTN H4D   /O 2
          /I 1
BTN H4D   /I 1 /O 2
```

Vont générer des erreurs de compilation.

5. Si vous utilisez les options /I, /O avec le bouton *S3,toutes les déclarations en position* /O seront généralement ignorées *par le « Compiler ».Généralement, car si vous utilisez la déclaration* S3_LOCK *(voir plus loin), alors vous pouvez générer des déclarations avec l'option* /O sur le bouton *S3. Si vous utilisez un bouton différent pour affecter S3, en utilisant la déclaration «* USE *Btn* AS SHIFTBTN » *(voir plus loin) ces remarques s'appliquent également.*

6. *Sion a écrit un fichier pour un joystick et une manette des gaz, mais seul le joystick est présent alors le « compiler »utilisera le code* /M *sur chaque bouton ou hat, et les options* /U, /D *seront ignorées.*

NOTES –COMMENT PREVENIR LE BLOCAGE DES COMMANDES

Voici des remarques techniques qui vont vous permettre de comprendre ce qu'il arrive lorsque vous changez l'état de S3(Appuyer ou relâcher et vice versa) et lorsqu'un bouton qui est programmé avec les options /I *et* /O *est sélectionné. Si la position du switch dogfight change, où l'état de S3 change, voilà ce que fait le contrôleur :*

1. *Le Cougar reconnaît le changement d'état*
2. *Le Cougar vérifie les boutons enclenchés*
3. *Le Cougar vérifie les boutons enclenchés afin de voir s'ils sont programmés différemment dans la nouvelle programmation.*
4. *Si elle existe, il déclenche l'exécution de la Macro.*
5. *Le Cougar est dans un nouvel état.*

3.5.2 Séparation des macros sur un bouton :

Attributs de déclaration

/T (Toggle) Répéter les différentes macros d'un bouton
/P, /R Appuyer (Press) et relâcher (Release) un bouton

3.5.2.1 /T – Barre d'attribut Toggle

Il est plus facile de comprendre l'action de cette commande (/T) à l'aide d'un exemple :

```
BTN S2 /T a
       /T b
       /T c
```

Soulignons que le bouton S2 est pressé 3 fois. La première fois on génère un "a". La fois suivante, un "b", et la dernière fois un "c". Si l'on appuie de nouveau, cela va produire un "a", et le cycle se répète, comme indiqué dans la macro.

On peut répéter jusqu'à 16 positions, incluant les barres d'attribut /U, /M, /D, /I et /O. Vous pouvez donc avoir :

BTN S2 /U /I *16 répétitions peuvent être insérées ici*
/O *16 répétitions peuvent être insérée ici*
/M /I *16 répétitions peuvent être insérée ici*
/O *16 répétitions peuvent être insérée ici*
/D /I *16 répétitions peuvent être insérée ici*
/O*16 répétitions peuvent être insérée ici*

NOTES

1. L'option /T n'est pas permise après les déclarations /P ou /R . *Donc si vous avez :*

Exemple 1. BTN TG1 /T /P a
/R b
/T c

Exemple 2. BTN TG1 /P /T a /T b
/R c

L'Exemple 1 est permis, mais 2 et 3vont générer une erreur de compilation.

Exemple 3. BTN TG1 /P a
/R /T b /T c

2. *Vous ne pouvez pas utiliser la déclaration* /T *sur T1 si vous avez utilisé la déclaration «* USE T1_SENSITIVITY *» dans votre fichier joystick (voir plus loin).*

3. *Vous ne pouvez pas utiliser la déclaration* /T *sur un hat du joystick, si vous avez utilisé la déclaration* USE *HatID*_SENSITIVITY *dans votre fichier joystick (voir plus loin).*

4. *Vous ne pouvez pas utiliser l'option* /T *avec une déclaration d'un axe digital (voir plus loin).*

5. *Vous ne pouvez pas utiliser l'option* /T *en programmant un état logique (voir plus loin).*

6. *Un seule barre d'attribut* /T *dans une ligne de commande, générera une erreur de compilation donc :*
BTN T4 /T a

N'est pas une expression permise.

7. *La déclaration /T peut être formulée sur une même ligne ou sur plusieurs :*

BTN S2 /T a /T b /T c

8. *Vous pouvez annuler cette instruction en utilisant l'expression* RESET_TOGGLES*, et vous pouvez modifier la mise en place en utilisant l'expression* REVERSE_TOGGLES*.*

9. *Vous pouvez utiliser d'autre barre d'attribut avec /T. Par exemple :*

BTN S2 /T a /T /H b   est une commande permise.

3.5.2.2 Annulation de la mise en place d'une commande /T

Si vous désirez annuler l'option /T vous pouvez très facilement le faire en utilisant :

Syntaxe de la commande
RESET_TOGGLES

Donc si vous avez une expression qui ressemble à ceci :

```
BTN S2 /I RESET_TOGGLES
       /O /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
```

et que vous appuyez plusieurs fois sur le bouton S2 afin, par exemple, de générer un '7'.Puis vous désirez revenir rapidement au début de la séquence. Vous pouvez le faire en appuyant sur le bouton S2 et le bouton S3 (afin d'activer l'option /I).
Appuyer sur le bouton S2 alors que le bouton S3 est enfoncé ne va générer aucun caractère. Vous devez appuyer sur S2 avec S3 relâché, alors vous générerez un '1' comme l'indique la commande ci-dessus.

NOTES

1. *Cette expression doit apparaître directement après un /I ou un /O ,.l'expemple suivant va générer une erreur :*

```
BTN S4   /U RESET_TOGGLES
         /M /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
         /D Some_macro
```

Voici une expression correcte :

```
BTN S4    /U Some_macro
          /M /I RESET_TOGGLES
          /O /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
          /D Some_macro
```

2. *L'expression* RESET_TOGGLES *sera seule sur sa ligne de commande*

3.5.2.3 Inverser la direction de la répétition

Si vous désirez inverser la direction de la répétition sur un bouton alors utilisez :

Syntaxe de la commande
REVERSE_TOGGLES

```
BTN S2 /I REVERSE_TOGGLES
       /O /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
```

Normalement en appuyant sur S2 vous pouvez générer les caractères allant de 1 à 0. Si vous appuyez sur S2 avec S3 enfoncé, alors vous allez appliquer la répétition des caractères en sens inverse.(0 puis 9 puis 8 etc.…)

NOTES

1. *Cette expression doit apparaître directement après un* /I *ou un* /O *,la commande suivante va générer une erreur :*

```
BTN S4 /U REVERSE_TOGGLES
       /M /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
       /D Some_macro
```

Voici l'expression correcte :

```
BTN S4 /U Some_macro
       /M /I /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
          /O REVERSE_TOGGLES
       /D Some_macro
```

2. *L'expression* REVERSE_TOGGLES *sera seule sur sa ligne de commande. Dans l'exemple suivant :*

```
BTN S4 /U Some_macro
       /M /I /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
          /O REVERSE_TOGGLES Macro_here_generates_error
       /D Some_macro
```

La macro placée après la commande REVERSE_TOGGLES générera une erreur de compilation.

3.5.2.4 /P, /R – Appuyer et relâcher (Press et Release)

La barre d'attribut /P aura pour effet d'exécuter la macro qui la suit lorsque l'on appuie sur le bouton ou le switch. La barre /R indique que la macro spécifiée sera exécutée lorsque le bouton ou le switch sera relâché.

Exemple : BTN S2 /P Chaff
/R Flare

Ici, lorsque l'on va appuyer sur le bouton S2, cela va avoir pour effet de générer la macro « Chaff » et lorsque le bouton sera relâché la macro « Flare » sera exécutée.

NOTES

La barre /R devra être utilisée en conjonction avec la barre /P. Si la barre /P n'est pas présente alors si on utilise la barre /R on obtiendra une erreur de compilation et vice versa.

NOTES

1. S*i vous utilisez ces barres d'attribut avec un « hat »,en conjonction avec une commande du hat en position milieu, alors la macro située après le /R sera générée en même temps que la déclaration BTN HxM .Alors:*

BTN H1U /P 1 /R 2
BTN H1M a

lorsque l'on appuiera et relâchera le hat1 on obtiendra
*"1" puis "**a 2**" ensemble*

2. *Si vous utilisez ces barres avec BTN HxM,* lorsque *le hat sera dans une position différente de la position milieu, on générera les caractères associés avec la déclaration HxM /R en même temps que ceux qui sont associés à H1U. donc :*

BTN H1U 1
BTN H1M /P a /R b

Génére lorsque l'on appuie et que l'on relâche HAT 1 :
*"**b 1**" simultanément puis un "a"*

3. *Nous avons également la possibilité de produire continuellement des caractères en utilisant les déclarations /P /R. Par exemple considérons :*

BTN S4 /P DLY(2000) KD (p)
/R KU (p)

En appuyant(garder appuyer) sur le bouton S4 , nous obtiendrons en premier lieu une temporisation de 2 secondes, puis cela émulera la touche "p"(pression continue) du clavier. Lors du relâchement du bouton S4, cela émulera l'action de relâcher la touche du clavier.

Mais que ce passe t il si vous relâchez le bouton S4 avant le délai de 2 secondes ?
La temporisation de 2 secondes est en cours, le bouton S4 étant relâché de façon prématurée, le premier code généré sera celui du relâchement de la touche « p ». Il sera généré avant le code d'enfoncement de la touche. Malheureusement, cela va donc provoquer le blocage de la touche 'p. Il existe 2 façons d'éviter ce problème. La première est d'utiliser et de programmer votre HOTAS de façon « logique ». Si vous avez connaissance d'un délai de 2 secondes, alors appliquez le ! Cela évitera les problèmes de blocage de touches. La deuxième façon est « d'encadrer » /P par < > *:*

BTN S4 /P **<** DLY(2000) KD (p) **>**
/R KU (p)

Cela aura pour effet de « forcer » l'exécution de la commande contenue entre les signes < >, puis l'attribut /R sera exécutée même si vous avez relâché votre doigt trop tôt. *Notons que toutes les macros de tous les boutons ne seront exécutées qu'après l'exécution des déclarations contenues entre < >.Soyez prudent lorsque vous les utilisez !*

3.5.3 Répétition ou non répétition des caractères :

Attributs de déclaration

/A Auto-repeat
/H Hold

3.5.3.1 Non répétition des caractères

Le comportement des macros et les déclarations d'un simple caractère sur un bouton ont changé depuis la conception de la syntaxe TM originale. Par défaut l'action des boutons, qu'ils désignent de simples caractères ou des macros, n'est pas répétée. Il y aura juste un caractère de générer. Pour ceux qui ont l'habitude

d'utiliser le TM HOTAS, cela se passe comme si il existait un /N devant chaque déclaration.

Par conséquent, l'option /N n'est plus utile – en fait, elle est ignorée par le « compiler », quand il en détecte une. Nous vous recommandons de les supprimer si elles sont présentes dans vos fichiers.

3.5.3.2 /A – Répétions automatique (Auto-Repeat)

La barre d'attribut /A a un effet exactement contraire. Elle oblige la répétition automatique d'un caractère ou d'une macro affecté à un bouton ou à une déclaration digitale jusqu'à ce que le bouton soit relâché.

BTN S2 /A Fire_Missiles

Dans ce cas, lorsque l'on appuie sur le bouton S2, la macro « Fire_Missiles » est répétée. C'est comme si vous aviez appuyé et relâché rapidement une touche sur votre clavier. Si d'autre boutons sont alors utilisés sur votre contrôleur, ils généreront également des caractères. En appuyant sur un autre bouton, vous n'interromprez pas la production de caractères générés par cette déclaration. Il faut noter que les macros suivant un /A peuvent comprendre de nombreuses sortes de caractères ou de déclarations comme, DLY, RPT etc...

3.5.3.3 /H –Maintenir (Hold)

BTN S4 /H b Rem Wheelbrakes

Une barre d'attribut /H peut être utilisée afin de simuler le maintien d'une touche enfoncée sur le clavier.(Voir *Note section suivante*). Par exemple de nombreux simulateurs vous demande de maintenir la touche "b" de votre clavier afin de pouvoir utiliser les freins. Dès que vous relâchez cette touche les freins sont relâchés.

D'autres exemples :

BTN S4 /H b Rem Wheelbrakes
BTN T6 /H Wheelbrakes
BTN T1 /H ALT F6

Il est possible de construire des commandes plus complexes à l'aide de cette option. Dans l'exemple qui suit, cette déclaration équivaut à générer un "c", suivi par un "f" (maintenu enfoncé sur le clavier) :

BTN S4 /H c f Rem 1 Chaff, loads of Flares

La déclaration qui suit est également intéressante à étudier :

BTN S4 /H {C f}

Il faut se rappeler que le "C" est en fait un "SHF c" et par conséquent si vous maintenez la touche « LSHF » enfoncée la touche "f" générée sera en fait un "F" majuscule. Il nous est donc impossible de reproduire un "f" minuscule alors que la touche SHIFT est maintenue enfoncée.

Dans l'exemple suivant, il est possible de sélectionner une arme et de maintenir la touche de mise à feux enfoncée jusqu'à son largage :

BTN S4 /H Select_Rockets DLY(600) Fire Rem Select secondary weapons then fire

NOTES

1. *Il faut noter un point important. Il existe une différence entre la barre d'attribut /H et /A : Normalement lorsque vous appuyez sur la touche "a" du clavier, cela produit un "a" et après un délai assez court une suite de "a" jusqu'à ce que vous relâchiez la touche* a *délai* aaaaaaaaaaaaaaaa)

   C'est la même chose en utilisant l'option /H :

   BTN T3 /H a

   Cette commande est l'exacte reproduction de l'exemple ci-dessus. Si nous voulons supprimer le délai entre le premier et le deuxième caractère, il nous faudra utiliser la barre d'attribut /A :

   BTN T2 /A b   *production de bbbbbbbbbbbbbbbbbb sans délai.*

2. *Le taux de répétition de chaque caractère généré avec l'option /A peut être modifié avec la commande* USE RATE (temps_ms). *Si celui-ci n'existe pas, le compiler va insérer une déclaration* USE RATE (0) et le taux de répétition sera celui du clavier par défaut.

3. *Si la barre d'attribut /H est suivie par de nombreuses macros ou une série de caractères, le seul caractère qui sera répété sera le dernier. Donc dans lea déclaration qui suit :*

   BTN S2 /H a b c

   Lorsque l'on appuie sur le bouton S2 et qu'il est maintenu enfoncé, alors un "a" et "b" seront générés avant que l'on obtienne une répétition de "c".

4. *Vous ne pouvez pas utiliser /H ou /A sur une position milieu d'un Hat, ou après la barre d'attribut /R.*

3.5.4 Règles des barres d'attribut et hiérarchie

Lorsque vous utilisez des barres d'attribut (ex, /T, /P, /U), vous devez savoir qu'il existe des règles élémentaires pour la mise en forme des déclarations et l'ordre dans lequel elles sont exprimées.

3.5.4.1 Règles des barres d'attribut

1. Elles doivent être placées après un code bouton (ex : BTN S2 /H)
2. Elles doivent être placées avant la première macro(ex :. BTN S2 /H Macro) excepté pour /T qui peut apparaître dans une déclaration plusieurs fois.
3. Il doit y avoir un seul espace avant et après une barre d'attribut (voir l 'exemple précédent)
4. Elles doivent apparaître dans un certain ordre lorsqu'elles sont plusieurs dans une déclaration (voir plus bas).
5. Les barres d'attribut /U /M /D doivent être sur des lignes différentes.
6. Les barres d'attribut /I /O ne doivent pas apparaître sur la même ligne.

3.5.4.2 Hiérarchie des barres d'attribut

Lorsque vous utilisez des barres d'attribut pour configurer un bouton ou un switch, il est important d'utiliser une hiérarchie appropriée.

1. Les barres /U /M /D, si elles sont présentes doivent précéder toutes les autres barres.
2. Les barres In/Out /I et /Oseront alors les prochaines.
3. La barre (/T) doit précéder les barres /P /R.
4. Les barres (/P et /R) sont toujours situées après celle-ci.
5. Et le /H et /A sont toujours placés en dernier.

Voyons maintenant quelques exemples :

```
BTN S4    /U  /I  macro6
          /O macro7
          /M /P macro8 /R macro9
          /D /T a
          /T /H b c
          /T /A d DLY(30) e DLY(30) f

BTN S2    /I  /T /P macro1 /R macro2
          /T  /P macro3 /R macro4
          /O  /H macro5
```

NOTES

Il n'existe pas de problèmes particuliers à laisser une déclaration vide. Vous n'avez pas besoin d'utiliser les caractères Nul comme vous deviez le faire avec la version précédente du langage « TM HOTAS ». Les déclarations qui suivent ne vont pas générer d'erreur :

```
BTN S1   /I Drop_Stores
         /O
```

3.6 Les déclarations de délai et de répétition

Attributs de déclaration DLY (Delay) RPT (Number)

« Delay » est une période en millième de seconde (1 seconde = 1000 millièmes de secondes), les valeurs comprises entre 0 et 82800000 sont acceptées. (Pour information, *82800000 millièmes de secondes équivalent à environ 23 heures !)*

« Number » est une valeur que l'on peut appliquer à une macro, elle peut être comprise entre 2 et 127.

Les exemples qui suivent vont nous permettre de comprendre le fonctionnement de ces déclarations.

3.6.1 Déclaration DLY( )

```
BTN T6 1 DLY(60) 1 DLY (60) 2 Rem Request Vector For Recovery TAW
```

Cette expression va générer un "1 1 2", avec 60 millisecondes de délai entre chaque caractères. En effet, les simulateurs de vols et autres jeux sont devenus si complexes qu'il est commun de voir une suite de caractères utilisée pour effectuer une action. Par exemple dans le célèbre Falcon 4 :

```
VectorToHomePlate = q DLY(60) q DLY(60) 6
```

Si la macro avait été créée sans la déclaration DLY comme ci-dessous,

```
VectorToHomePlate = q q 6
```

il y a de grandes chances pour que le simulateur, qui est par ailleurs très occupé par ses propres routines, ne puisse pas prendre en compte ces trois caractères du fait de la vitesse du contrôleur. Dans ce cas si l'on place une déclaration DLY entre chaque caractère, cela va ralentir leur production et ainsi permettre leur prise en compte par le simulateur. Cela devient un peu plus complexe lorsqu'une déclaration USE RATE est présente.(Voir plus loin). Une déclaration USE RATE (time)va déterminer la rapidité du contrôleur.

Donc :

```
USE RATE (60)
BTN S1 q q 6
```

Est exactement la même chose que :

```
USE RATE (0)
BTN S1 q DLY(60) q DLY(60) 6
```

Par défaut le compiler utilise la déclaration USE RATE(0) si elle n'existe pas dans le fichier .

Lorsque vous utilisez une déclaration DLY, vous n'avez pas besoin de garder le doigt sur le bouton jusqu'à la complète exécution de la déclaration. Par exemple :

```
BTN S2 h DLY (2000) e DLY (2000) l DLY (2000) l DLY (2000) o
```

Donne un "hello", avec 2 secondes de pause entre chaque caractères. Vous n'avez pas besoin de maintenir votre doigt sur le bouton S2 pour que la commande soit exécuté en entier. Il est intéressant que si vous appuyer une deuxième fois sur le bouton S2 à 2 secondes d'intervalle, vous obtiendrez :

"hheellllloo"

C'est une des caractéristique du HOTAS Cougar – il est capable d'effectuer plusieurs tâches en parallèle. Pour vous mettre à l'abris d'erreurs éventuelles, faites attention à la manière dont vous utilisez le Cougar, ou alors insérer vos déclarations entre des parenthèses < >.

3.6.2 Déclaration RPT( )

```
BTN S2 RPT(6) c  Rem 6 chaffs s.v.p… avec ça cela devrait aller !
```

En utilisant la commande RPT dans une déclaration, les caractères ou macros qui suivent cette commande seront répétés un nombre « nnn » de fois (RPT(nnn)). L'objet répété sera celui qui se trouve **immédiatement** après la commande RPT. De

plus, cet objet pourra être un caractère, un groupe de caractères ou des commandes situées entre des parenthèses.

Dans l'exemple ci-dessus, en appuyant sur le bouton S2, on génère 6 "c" à vitesses (taux) définie par la commande USE RATE(x) dans le fichier.

Exemples supplémentaires :

BTN S1 RPT(10) a b

10 caractères "a" suivis par un caractère "b".
BTN S1 RPT(10) (a b)

10 caractères "a b".

BTN S1 /A RPT(10) (a DLY(60)) DLY(2000)

Cela va générer 10 "a" avec 60 millisecondes de délai entre chaque caractères, puis un délai de 2 secondes, et finalement toute la déclaration pourra se répéter.

Le nombre de répétitions RPT () peut être peut être « imbriqué » : par exemple :

BTN S1 RPT(10) a RPT(10) b

Est une déclaration valide,

BTN S1 RPT(10) (a RPT(10) b)

Et celle-ci est également valide.

Si vous avez une macro comme la suivante :

Macro1 = a b c

Et une déclaration comme ci-dessous :

BTN S2 RPT (3) Macro1

Alors quand vous appuierez sur S2, vous obtiendrez :

a a a b c

pour éviter ceci, soit vous insérez les macros entre des parenthèses ou vous insérez les caractères désirés entre des parenthèses :

BTN S2 RPT (3) (Macro1) ou

BTN S2 RPT (3) Macro1 dans ce cas :

Macro1 = (a b c)

3.7 Regroupement de caractères – utilisation des parenthèses

Attributs de déclaration

BTN T2 (a b c)
BTN T3 {a b c}
BTN T4 /P <a b c>
/R d

Avant de nous intéresser à l'utilisation des parenthèses, nous devons respecter une règle très importante : Les parenthèses **( ) { }** and **< >** sont réservées à l'usage de la programmation des déclarations.
Par conséquent vous ne pouvez pas les assigner directement à un bouton ou une macro, pour ce faire vous devez utiliser la déclaration majuscule "*shitf*" :

(= SHF **9**
) = SHF **0**
{ = SHF **[**
} = SHF **]**
< = SHF **,**
> = SHF **.**

Donc la déclaration d'une macro :

Left_ToeBrake = <

Va générer une erreur de compilation, la déclaration exacte sera :

Left_ToeBrake = SHF ,

3.7.1 Les parenthèses ( )

Dans la syntaxe TM les parenthèses sont souvent utilisées : cf. DLY ( ), RPT ( ), USB ( ), KU ( ), pour regrouper des déclarations etc. Nous avons vu plus haut les exemples concernant RPT et DLY :

BTN S1 /A RPT(10) (a DLY(60)) DLY(2000)

Dans la première syntaxe du langage Trustmaster, lorsque l'on insérait des caractères ou des macros entre des parenthèses, on obligeait la répétition de ces caractères. Ce n'est plus le cas avec le Hotas Cougar, comme nous l'avons vu avec l'option /A modifier. Maintenant les parenthèses sont utilisées pour regrouper des caractères ou des macros dans une déclaration.

Notez que si des parenthèses sont utilisées dans des déclarations TM, le compiler interprète ces déclarations comme étant les mêmes : DLY (20), DLY(20), DLY (20) etc.

Un dernier point avant de quitter ce chapitre, si vous voulez générer "(" ou ")" dans votre fichier joystick, alors n'utilisez pas ces caractères directement : cf.

Macro1 = (
Macro2 =)

Les parenthèses et autre sigles comme (< > { }) sont “réservés”, ils sont utilisés dans de nombreuses déclarations et affectent grandement le sens de ces déclarations.

Don si vous avez :

BTN S1 Macro1

Lors de la compilation, cela ne va pas générer un caractère"(" character. A la place utilisez les commandes ci-dessus, dans ce cas nous aurons :

Macro1 = SHF 9
Macro2 = SHF 0

Voyez également le chapitre Macros. Rappelez vous également que vous pouvez utiliser le programme Korgy, afin de vous assurer que vous utiliser la bonne syntaxe pour générer des commandes ou des caractères.

3.7.2 Les parenthèses « crochet » { } (*Curly brackets)*

L'autre façon d'effectuer un regroupement est d'utiliser les parenthèses « crochet » (Curly Brackets). Elles permettent de générer un groupe de caractères comme si l'on avait appuyé sur les touches correspondantes en même temps. Le groupe compris entre ces parenthèses est traité comme une entité. Par exemple :

BTN S4 {a b c}

Correspond à une pression sur "a", "b", "c", et un relâchement de "a", "b","c" comme si vous aviez pressé puis maintenue les touches les une après les autres, puis vous les auriez relâché dans le même ordre.(Pensez à ce que vous faites lorsque vous devez utiliser la combinaison de touches CTRL ALT DEL)

ces parenthèses peuvent être utilisées avec la barre d'attribut /H, dans ce cas toutes les touches comprises entre ces crochets seront maintenu enfoncées jusqu'à ce que le bouton soit relâché. Par exemple :

BTN S4 /H {CTL ALT DEL} est une combinaison parfaitement acceptable.

NOTES

1. *Vous ne pouvez pas utiliser la commande DLY entre ces parenthèses.*

2. *L'utilisation des parenthèses { } diffère suivant la version USB ou port jeux des TM Hotas. Dans la version précédente (port jeux), les caractères contenus entre les parenthèses { } étaient générés dans l'ordre de leur apparition, donc avec :*

BTN S4 {a b c}

On générai : appuyer sur "a", appuyer sur "b", appuyer sur "c", relâcher "a",relâcher "b", relâcher "c".
Avec le port USB, bien que dans ce cas l'ordre des caractères soit conforme aux références USB, les caractères seront générés dans l'ordre alphabétique, nous aurons donc :

BTN S4 {c b a}

Cette ligne correspondra à : appuyer sur "a", appuyer sur "b", appuyer sur "c", relâcher "a", relâcher "b", relâcher "c. En réalité, les caractères seront tous générés en même temps. Cela équivaux à appuyer sur les touches a, b, c du clavier en même temps et relâcher ces touches de la même façon. Si vous désirez vraiment les placés dans un ordre spécifique, alors vous serez obligé d'utiliser les déclarations KD *et* KU*, comme ci-dessous :*

BTN S4 KD(c) KD(b) KD(a) KU(c) KU(b) KU(a)

33.7.3 Les parenthèses obliques <> (*Angle brackets*)

Les parenthèses obliques font parties de la nouvelle syntaxe du HOTAS Cougar. Ces parenthèses vont forcer le contrôleur à terminer toutes les déclarations contenues entre celles-ci avant qu'il ne puisse en démarrer d'autre. Par exemple considérons l'exemple suivant :

BTN H1D q DLY(60) q DLY(60) 6 Rem Vector for Homeplate in Falcon 4

Il serai dommage d'appuyer sur un autre bouton dans le même temps, cela pourrai changer le sens de cette demande dans Falcon 4, alors en insérant < > :

BTN H1D <q DLY(60) q DLY(60) 6> Rem Vector for Homeplate

Lorsque l'on appuie sur le Hat 1, on la complète exécution de la déclaration avant qu'une autre puisse être interprétée. Cela peut être très utile pour éviter les érreurs de « touché » :

BTN T4 /P < DLY(2000) KD (b) >
/R KU (b)

En appuyant sur le bouton T4, la barre /P vous donne la garantie que la déclaration sera complètement exécutée, même si vous relâchez le bouton de façon prématurée. Donc le caractère "KU (b)" ne sera généré qu'après que l'on a appuyé sur la touche "b". Si ce n'était pas le cas, alors l'action "KU (b)" sera généré dès que T4 sera relâché

NOTES

1. *Vous ne pouvez pas utiliser < > à l'intérieur de { }, donc :*

BTN S2 { < a b > }   *générera une erreur de compilation.*

2. *Vous ne pouvez pas utiliser les caractères < > en « cascade », donc :*

BTN T4 < < a b > >   *générera une erreur de compilation.*

3. *Les parenthèses < > n'ont pas besoin d'encadrer une déclaration complète, donc :*

BTN S1 a b <c d > e f   *est une expression correcte.*

4. S*i vous rencontrez une déclaration comme celle ci-dessous :*

BTN S2 /H <a b c>

Dès que les caractères “a” et “b” ont été produits, la déclaration exécute l’instruction de maintenir la touche “c” enfoncée. Ceci est possible grâce à l’utilisation de la barre d’attribut /H.

5. *Avec l’utilisation des parenthèses < >, si une autre déclaration est exécutée à ce moment, il n’y a pas d’interruption, elle continue de l’être. Donc, si vous êtes en train d’utiliser un bouton avec* /H *autre part, cela va continuer à fonctionner jusqu’à ce que vous appuyer sur le bouton associé à la déclaration contenant la déclaration < >. Tous les autres boutons qui seront enfoncés, seront alors mis en mémoire, les actions seront exécutées lorsque la déclaration contenue entre les < > sera complètement achevée. Soyez prudent lors de l’utilisation de ces parenthèses et de plus si vous utiliser* DLY ( ) *à l’intérieur de celles-ci, il est possible que vous bloquiez le système. (Voir la section dépannage plus loin)*

NOTES

Reprenons l’exemple vu plus haut :

BTN S4 KD(c) KD(b) KD(a) KU(c) KU(b) KU(a)

Cette déclaration à pour effet de fractionner chaque expression KeyDown *et* KeyUp en *6 blocs différents. Si il existe une commande* RATE *par défaut, l’utilisateur voudra sans doute que les déclarations* KeyUp *se produisent en même temps, et non à la vitesse définie par la commande* RATE . *Cela peut être fait de la façon suivante :*

BTN S4 KD(c) KD(b) KD(a) KU({c b a})

Ains, les déclarations KeyUp *sont regroupées en un seul bloc et par conséquent se produisent plus rapidement.*

3.8 Utilisation et définition des boutons DirectX

Si vous avez utilisez un joystick simple, avec juste une gâchette, vous avez sans doute remarqué que cette gâchette vous permettait de larguer vos missiles ou de faire feu avec vos canons. Cela marche effectivement, non pas parce que vous avez programmé votre joystick mais parce que Windows signale au jeu que le joystick à une gâchette. Cette gâchette est juste un bouton, un tel bouton est appelé bouton DirectX – un bouton dont la fonction est assignée par le jeu.

Déclaration de Configuration

USE *button_identifier* or *logical_flag* AS DX*n*

Syntaxe

BTN *button_identifier* DX*n*

où:

button identifier est H1U, T6, S2 etc.
logical flag de X1 à X32 (*expliqué plus loin dans le guide de référence*)
n de 1 à 28

Le HOTAS Cougar est constitué de 10 axes (avec le nouveau palonnier), 28 boutons, et un POV HAT (Hat 1). Lorsque le Cougar est en mode Windows, les boutons peuvent être assignés par le jeu ou le simulateur de vol. Cela peut se faire car Windows informe le simulateur des capacités du système. Quand vous programmez un fichier joystick vous insérez des déclarations qui vont permettre d'informer le simulateur de la configuration du joystick.

Par défaut en mode programmation, aucun bouton ne pourra être utilisé comme un bouton DirectX. Donc si vous désirez le faire vous devez en informer le simulateur. Le bouton le plus couramment utilisé comme un bouton DirectX est la gâchette :

USE TG1 AS DX1

Vous n'avez pas besoin de faire autre chose – La gâchette sera reconnu par le simulateur et une fonction lui sera attribuée , habituellement la mise à feu ou le tir canon voici d'autres exemples :

USE H1U AS DX2
USE X4 AS DX3 Rem assigned to a logical flag - *see later notes*
USE T4 AS DX5

Vous pouvez également programmer toutes les déclarations DXn à une déclaration de bouton, et faire encore beaucoup de choses très astucieuses :

BTN S2 /H a DLY(2000) **DX2**

Dans cet exemple, lorsque vous maintenez le bouton S2 enfoncé, un caractère "a" sera généré, suivi du DX2 avec deux secondes de délai. Dans ce cas nous avons presque défini le bouton S2 comme le bouton DX2 parce que nous n'avons pas utilisé la déclaration USE. Le terme presque est employé car l'expression précédente n'a pas la même signification que celle qui suit :

USE S2 AS DX2

Elle a pour effet de générer le bouton DX2 dès que le bouton S2 est enfoncé. Dans notre exemple le DX2 était généré uniquement après la complète exécution de "a DLY(2000)".

3.8.1 USE ALL_DIRECTX_BUTTONS

Nous avons vu comment attribuer individuellement des boutons comme boutons DirectX. C'est le bon moment pour vous présenter l'instruction, USE ALL_DIRECTX_BUTTONS.

Déclaration de configuration:

USE ALL_DIRECTX_BUTTONS

Cela attribue tous les boutons comme boutons DirectX. Ainsi un fichier contenant cette instruction devrait attribuer tous boutons non programmés comme boutons DirectX, bien que les courbes seront toujours modifiables, la souris active, etc. Notez que les options que vous aurez paramétré avec Foxy seront ignorées.

Ainsi vous pouvez avoir un fichier assez simple :

```
Rem Paramétrage de tous boutons comme boutons DirectX
USE ALL_DIRECTX_BUTTONS
Rem Attribution de la souris au microstick - voir notes dans le livre de ref
USE MTYPE A3
Rem Et attribution du chapeau Hat1 sur le joystick comme chapeau POV
USE HAT1 AS POV
```

Ca devrait vous donner après transfert, un joystick et throttle ayant le Hat 1 comme POV, les boutons en DirectX, et la souris sur le microstick. Vous pouvez alors commencer à ajouter des instructions et faire progressivement un fichier.

NOTES

1. Les syntaxes remplaçant le PORT Bx sont les instructions de l'*HOTAS d'origine.*
2. *Les attributions des boutons DirectX pour le Cougar en fenêtre Windows sont :*

DX	BTN ID	DX	BTN ID	DX	BTN ID	DX	BTN ID
1	*TG1*	*8*	*H2R*	*15*	*H4U*	*22*	*T4*
2	*S2*	*9*	*H2D*	*16*	*H4R*	*23*	*T5*
3	*S3*	*10*	*H2L*	*17*	*H4D*	*24*	*T6*
4	*S4*	*11*	*H3U*	*18*	*H4L*	*25*	*T7*

5	*S1*	*12*	*H3R*	*19*	*T1*	*26*	*T8*
6	*TG2*	*13*	*H3D*	*20*	*T3*	*27*	*T9*
7	*H2U*	*14*	*H3L*	*21*	*T2*	*28*	*T10*

Avec le Hat1 *définit par défaut comme chapeau POV (Point de vue).*

3. *Si vous utilisez l'instruction* USE ALL_DIRECTX_BUTTONS *dans votre fichier, et que vous avez programmé un des boutons de votre contrôleur, alors il ne sera pas paramétrer comme bouton DirectX. La façon dont le compilateur convertit une instruction* USE ALL_DIRECTX_BUTTONS *est d'attribuer les boutons DirectX par défaut comme suit :*

BTN TG1 /H DX1
BTN S2 /H DX2

etc. Maintenant, si dans votre fichier, il se trouve une instruction définie pour le BTN S2, alors il ne paramètrera pas ce bouton comme bouton DirectX. Ceci est toutefois une bonne façon d'avoir la majorité de vos boutons attribuables dans un jeu, mais vous permettant aussi de programmer certains d'entre eux pour vos propres besoins.

4. *Si vous avez à utiliser une instruction* USE ALL_DIRECTX_BUTTONS*, et une combinaison de* USE MTYPE *et de* USE HATn AS POV (*voir notes plus loin*), *alors ces instructions **doivent** venir **avant** toutes instructions de bouton, ou le Compilateur générera une erreur. Souvenez que vous devrez toujours essayer et structurer votre fichier pour que les instructions apparaissent avant toutes instructions d'axes ou de boutons.*

5. *L'instruction* USE MTYPE*, (dont nous parlerons plus tard dans le livre de référence), peut attribuer les boutons souris gauche et droit aux T1 et T6 sur le throttle dépendamment du type d'instruction* MTYPE *que vous insérerez. Si vous devez avoir une instruction* USE MTYPE *avec une instruction* USE ALL_DIRECTX_BUTTONS*, alors si l'instruction* MTYPE *attribue des boutons comme bouton souris, ils ne seront pas désignés comme boutons DirectX.*

MTYPE (A1 *à* A5) *attribue les boutons souris comme suit :*

A1: *T1* = *Bouton souris gauche*, *T6* = *Bouton souris droit*
A2: *T1* = *Bouton souris droit*, *T6* = *Bouton souris gauche*
A3: *T1* = *Bouton souris gauche*
A4: *T6* = *Bouton souris gauche*
A5: *N'attribue aucun bouton souris.*

6. *Si vous choisissez un HAT différent comme contrôleur POV, (nous couvrirons ceci plus tard dans le livre de référence) par exemple:*

USE HAT3 AS POV

Alors le compilateur n'attribuera aucun bouton DirectX à ces positions.

NOTES AVANCEES

1. *Si vous prenez l'exemple utilisé auparavant :*

USE TG1 AS DX1

Alors ce que le compilateur fera, c'est de le traduire en l'instruction suivante :

```
BTN TG1   /P KD (DX1)
          /R KU (DX1)
```

Sachant que l'on peut utiliser les syntaxes KD *et* KU *pour séparer les parties* KeyDown *et* KeyUp *des boutons DirectX, voyons un exemple amusant :*

```
BTN TG1   /I   /A KD (DX1) DLY(50) KU (DX1) DLY (200)
          /O /H DX1
```

Maintenant si dans votre sim, le canon est attribué au bouton DirectX 1, alors avec l'instruction ci dessus, avec S3 out, le canon devrait faire feu, et avec S3 in, le canon devrait faire feu de façon intermittente.

3.9 Utilisation des codes KD, KU et USB

Il y a des occasions où vous allez vouloir plus de contrôle sur KeyDown et KeyUp, ou vouloir être capable d'envoyer un code direct qui définit une clé spéciale – Seulement pour les claviers non US. Ceci peut être possible avec :

Commande, syntaxe
KD(*Caractères Clavier/Boutons DX/Boutons souris*) KU(*Caractères Clavier/Boutons DX/Boutons souris*) USB(*Key_eventHID code*)

3.9.1 KD, KU

Toutes ces instructions sont désignées pour prévoir un contrôle de ce que vous voulez faire quand vous appuyez sur une touche clavier. Et évidemment quand vous appuyez sur une touche, vous appuyez sur **K**ey **D**own (**KD**) et alors en enlevant le doigt, vous permettez à la touche de remonter (**KU**). Parfois vous

voudrez être capable de programmer d'autres caractères pour être produit entre 2. Et vous pouvez le faire comme suit :

```
BTN H1U KD(UARROW) DLY(20) KU(UARROW)
```

Dans cet exemple, relâché le HAT1 revient à appuyer sur la touche haut pendant 20 ms, et à la relâcher. KD et KU peuvent être utilisé sur toutes les touches – vous devez juste utiliser la syntaxe TM correcte.

Il est aussi possible de combiner des touches contenant les instructions KD, KU.

```
BTN T4 KD(a b c) DLY(20) KU(a b c)
```

Vous pouvez aussi utiliser KD et KU sur les boutons DirectX, boutons souris et drapeaux logiques (*drapeaux logiques sont traités plus loin dans ce livre*). Par exemple avec le bouton souris gauche (MOUSE_LB) :

```
BTN T6 KD(MOUSE_LB) DLY(2000) KU (MOUSE_LB)
```

L'appui sur le bouton T6 imite l'appui sur le bouton souris gauche pendant 2 secondes et relâché ensuite.

3.9.2 Programmation USB

Il est possible d'être capable d'envoyer le code USB actuel pour soutenir une syntaxe qui n'est pas supportée dans les syntaxes par défaut TM. Ceci peut être utile pour les dispositions des claviers non US. Les codes USB sont fournis dans l'annexe 3 à la fin de ce livre et chaque code de touche est précédé soit par "D" pour un évènements KeyDown, soit par "U" pour KeyUp. Par exemple :

```
BTN T3 /P USB (D51) /R USB (U51) Rem 'Down arrow'
BTN T4 USB (DE1 D04 UE1 U04) Rem 'Shift a'
```

Les codes USB dans ces exemples produisent des appuis de touches et le font en se plaçant dans la structure mémoire dans des frames différents. Si vous voulez produire ces appuis de touches pour qu'ils se fassent au même moment, alors introduisez les avec les crochets { }. Par exemple :

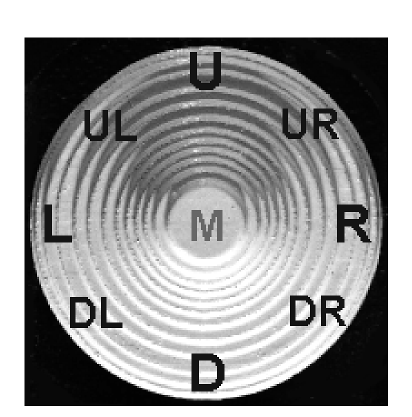
```
BTN T4 USB (DE1 D04) DLY (2000) USB ({UE1 U04}) Rem 'Shift a'
```

Entraîne le relâchement du Shift gauche et de la touche 'a' en même temps. Je devrais préciser que la différence de temps entre les frames est très petite – approximativement 30 ms, donc vous ne verrez probablement pas la différence.

4. Programmation des HAT

4.1 Programmation des Joystick HAT

4.1.1 Positions programmables sur un hat



Les Hat ont 9 positions programmables, quoiqu'en général, vous programmerez seulement les 4 directions principales. Pour le HAT 1 par exemple, ce serait :

BTN H1U Look_up
BTN H1R Look_right
BTN H1D Look_down
BTN H1L Look_left

Mais les positions de coin sont aussi programmables :

BTN H1UL View_UL
BTN H1UR View_UR
BTN H1DL View_DL
BTN H1DR View_DR



Ainsi que la position centrale :

BTN H1M View_forward

Il est important de noter que toutes ses positions programmables sont indépendantes et que les positions de coin ne sont pas par défaut un produit de ce qui est programmé sur les positions des autres cotés. Donc si je programme les positions principales haut, bas, gauche et droite pour être les touches 8, 2, 4 et 6 du pave numérique, alors l'appui du hat en position UR ne va pas aboutir au maintien des touches 8 et 6, ou de produire un 9. Ca ne fera rien du tout. Bien sur, cela dépends de votre capacité à déplacer le chapeau dans la position de coin directement.

4.1.2 Chapeau 4-voies contre 8-voies : USE HatID FORCED_CORNERS

Si je voulais forcer les positions de coin pour produire une combinaison des leurs positions opposées, alors je pourrais utiliser l'instruction :

Déclaration de configuration:

USE *HatID* FORCED_CORNERS

où: *HatID* est soit HAT1, HAT2, HAT3, ou HAT4
e.g. USE HAT1 FORCED_CORNERS

(*Oh, à propos, il y a une instruction que vous pouvez employer pour le faire plus facile pour utiliser les positions de coin plus sûrement : voir USE HatID_SENSITIVITY(nnnn) plus loin dans ce livre de référence*)

Un HAT peut être attribué afin de le programmer de façon normale, ou comme souris, POV (Contrôle de Point de Vue), touches fléchées, ou comme touches du pavé numérique. Il y a 4 instructions spéciales qui peuvent être utilisé pour paramétrer un hat pour différentes utilisations. Ce sont :

Déclaration de configuration:

USE *HatID* AS MOUSE (rate) [- *optional modifiers*]
USE *HatID* AS POV [- *optional modifiers*]
USE *HatID* AS ARROWKEYS [- *optional modifiers*]
USE *HatID* AS KEYPAD [- *optional modifiers*]

Où :

HatID est soit HAT1, HAT2, HAT3, HAT4, ou RADIOSWITCH

(*Le RADIOSWITCH bien qu'il ne ressemble pas à un HAT, est constitué des boutons T2 (Haut), T3 (Bas), T5(Gauche), T4 (Droite). Il diffère légèrement des 4 hats normaux dans le fait qu'il contient déjà l'instruction FORCED_CORNERS.*)

Rate est 1 à 127 et s'applique à l'instruction USE *HatID* AS MOUSE (rate).

[-*optional modifiers*] peut être utilisé avec ces instructions pour modifier le fonctionnement du hat. Ils consistent en :

REVERSE_UD, REVERSE_LR, FORCED_CORNERS, NOHOLD, KP5.

Notez que tous ces attributs optionnels ne peuvent pas être utilisé avec toutes les instructions de chapeau – voir la section correspondante plus bas, ou utiliser juste Foxy's Composer.

4.1.3 Contrôler la souris avec un HAT.

Déclaration de configuration :

USE *HatID* AS MOUSE (rate) [- *optional modifiers*]

Rate : - Vitesse de la souris comprise entre 1 et 127
Attributs Optionnels autorisés : REVERSE_UD, REVERSE_LR

e.g. USE HAT1 AS MOUSE (2)

Tout simplement, ceci attribue le contrôle de la souris sur le HAT 1. La valeur entre crochets, dans ce 2e cas, détermine le taux auquel la souris se déplace à travers l'écran – une valeur faible entraîne une souris paresseuse, et une valeur élevée entraîne une souris vive. Si le hat est déplacé dans une position de coin, alors la souris se déplacera en diagonale.

Si nous voulons inverser les directions Haut et Bas de la souris, nous pouvons le faire ainsi :

USE HAT1 AS MOUSE (2) - REVERSE_UD

Et de même pour inverser les directions droite-gauche, nous pouvons faire ainsi :

USE HAT1 AS MOUSE (2) - REVERSE_LR

Vous pouvez utiliser les deux inversions REVERSE_*types* ensemble si vous voulez. Ainsi :

USE HAT1 AS MOUSE (2) - REVERSE_UD, REVERSE_LR

Est une instruction parfaitement valide.

NOTES

Il est aussi possible d'avoir la souris contrôlée par chapeau, et en même temps par le microstick, ou n'importe où ailleurs. Il est aussi possible d'avoir un contrôle souris du chapeau à une position donnée au switch dogfight (/U, /M, /D*) sur le throttle, et/ou la position S3 (*/I, /O*). Ceci est expliqué plus en détails dans la*

section de programmation souris : Comprendre le périphérique souris et le Microstick.

4.1.4 Paramétrer un HAT comme Point de Vue (POV) HAT

Déclaration de configuration :

USE *HatID* AS POV [- *optional modifiers*]

Attributs optionnels autorisés : REVERSE_UD, REVERSE_LR

e.g. USE HAT3 AS POV

Quand vous utilisez le contrôleur en mode Windows, ou si vous n'avez pas d'instruction HAT 1 (BTN H1x) dans votre fichier, le HAT 1 par défaut opérera comme un POV HAT standard. Un POV HAT est un HAT 8-voies spécial qui peut être contrôlé par le sim dans beaucoup d'autres. Par exemple, dans Falcon 4, si vous ne programmez pas le HAT 1, il se comportera comme un POV HAT pour le contrôle de vue. De même dans les instructions précédentes, vous pouvez inverser les directions haut/bas et droite/gauche des POV avec :

USE HAT4 AS POV - REVERSE_UD
USE HAT1 AS POV - REVERSE_LR
USE HAT3 AS POV - REVERSE_UD, REVERSE_LR

Vous pouvez aussi programmer les postions POV en utilisant POVU, POVD etc, même si vous avez attribué un hat comme POV. Vous pouvez programmer n'importe quelle position que vous voulez (*et oublier celle que vous ne voulez pas*). C'est comme les boutons DX - Ils sont là, mais vous avez besoin de les programmer pour les rendre actifs. Aussi longtemps que le matériel détecte un POV présent sur le type de poignée que vous utilisez, vous pouvez programmer les positions POV. Voir les Notes de la section ci dessous.

NOTES

Nous avons comment c'est facile d'attribuer un hat comme contrôleur POV. Mais cela vaut aussi la peine de préciser que les positions du POV peuvent être programmé directement dans un fichier. La syntaxe pour les positions du POV sont identiques aux positions sur un hat normal. Ce sont :

POVU, POVD, POVL, POVR, POVUL, POVDL, POVUR, POVDR

Et vous devrez programmer comme suit :

BTN T5 POVL
BTN T4 POVR

4.1.5 Utiliser un HAT pour émuler les touches flèches du clavier

Déclaration de configuration:

USE *HatID* AS ARROWKEYS [- *optional modifiers*]

Attributs optionnels autorisés : REVERSE_UD, REVERSE_LR, NOHOLD

e.g. USE HAT2 AS ARROWKEYS

Il est très courant dans les sim de vol d'être capable de programmer les touches fléchées sur un hat, et c'est exactement l'utilité de ces instructions. Les touches fléchées seront désactivées pendant tout l'appui du hat. De même pour les instructions précédentes, vous pouvez aussi inverser les directions haut/bas et gauche/droite des touches fléchées avec :

USE HAT3 AS ARROWKEYS - REVERSE_UD
USE HAT4 AS ARROWKEYS - REVERSE_LR
USE HAT1 AS ARROWKEYS - REVERSE_UD, REVERSE_LR

Notez que si vous déplacez le hat dans une position de coin, alors le hat activera les touches fléchées de chaque coté de cette position – Il a son propre modificateur FORCED_CORNERS intégré.
Si vous ne désirez pas désactiver les touches fléchées, utilisez le modificateur NOHOLD avec cette instruction, comme ceci :

USE HAT3 AS ARROWKEYS - NOHOLD

Ceci reproduira une seule touche fléchée quand le chapeau sera déplacé dans les différentes positions. Ainsi avec l'autre modificateur optionnel, il pourra etre utilisé en rapport avec eux, comme ceci :

USE HAT1 AS ARROWKEYS - REVERSE_UD, REVERSE_LR, NOHOLD

4.1.6 Utiliser un HAT pour émuler le pave numérique

Déclaration de configuration :

USE *HatID* AS KEYPAD [- *optional modifiers*]

Attributs optionnels autorisés :
REVERSE_UD, REVERSE_LR, FORCED_CORNERS, NOHOLD, KP5

eg. USE HAT4 AS KEYPAD
Il est aussi courant de vouloir émuler le pave numérique avec un HAT. Avec les instructions précédentes donc, HAT 4 produira des nombres du pave numérique, avec les positions de coin du hat (UL UR, DL, DR) générant respectivement les touches "7 9 1 3".

Le problème avec le pavé numérique, c'est que son comportement varie énormément d'un simu ou jeu à l'autre. Qui plus est, les simus se comportent de façon différente si le Num LOCK est activé, ou ils peuvent l'obliger à être ON ou OFF. Dans la plupart des cas, ça fonctionne si on traite le pave numérique comme un simple générateur de chiffre.

De la même façon que dans les instruction précédentes, vous pouvez inverser les directions haut/bas et gauche/droite des touches fléchées avec :

```
USE HAT1 AS KEYPAD - REVERSE_UD
USE HAT2 AS KEYPAD - REVERSE_LR
USE HAT3 AS KEYPAD - REVERSE_UD, REVERSE_LR
```

Nous avons dit que l'instruction active les touches du pave numérique. Bien sur, il y a une touche manquante, et c'est la touche 5 (KP5). Certains simus attribuent le «5» pour centrer quelque chose d'utile, et vous pouvez programmer le compilateur pour associer la touche KP5 à la position centrale du hat, avec :

```
USE HAT4 AS KEYPAD - KP5
```

Nous pouvons aussi forcer les positions de coin :

```
USE HAT4 AS KEYPAD - FORCED_CORNERS
```

Qui commande le HAT 4 à se comporter ainsi si il est poussé dans la position de coin UR , et de produire à la place d'un "KP9", un "KP8" et un "KP6" ensemble, ie. Les caractères Haut et Droit attribués.
Ceci n'a pas beaucoup de sens si on considère que le hat n'est qu'un générateur de chiffres, mais interprétés correctement dans certains simus et jeux.

Si vous ne voulez pas que le pavé numérique soit activé, utilisez le modificateur NOHOLD avec cette instruction, comme suit :

```
USE HAT3 AS KEYPAD – NOHOLD
```

Cela produira une seule touche du pave numérique quand le hat sera déplacé dans ses différentes positions.

Vous pouvez combiner tous ces attributs optionnels si vous le voulez :

USE HAT4 AS KEYPAD - REVERSE_UD, REVERSE_LR, FORCED_CORNERS, NOHOLD, KP5

4.1.7 Comment le Compilateur transforme l'instruction USE HatID AS

Cette section est pour les utilisateurs expérimentés, avec un énorme appétit du détail !

Vous devez avoir lu les autres sections de ce livre pour comprendre les notes détaillées suivantes, mais pour ceux qui aiment ce qui est technique, je vous parle brièvement de la façon dont le compilateur transforme actuellement les différentes instructions USE HAT*n* AS *quelquechose_utile*. Nous commencerons avec l'instruction USE HATx AS MOUSE.

USE HAT1 AS MOUSE (2) est transformé par le compilateur en :

USE HAT1 FORCED_CORNERS
BTN H1U /P MSY(2-) /R MSY(2+)
BTN H1R /P MSX (2+) /R MSX (2-)
BTN H1D /P MSY (2+) /R MSY (2-)
BTN H1L /P MSX (2-) /R MSX (2+)

De même, le Compilateur change : USE HAT1 AS MOUSE (2) -REVERSE_UD

En : USE HAT1 FORCED_CORNERS
BTN H1U /P MSY(**2+**) /R MSY(**2-**)
BTN H1R /P MSX (2+) /R MSX (2-)
BTN H1D /P MSY (**2-**) /R MSY (**2+**)
BTN H1L /P MSX (2-) /R MSX (2+)

Et finalement : USE HAT1 AS MOUSE (2) - REVERSE_UD, REVERSE_LR

En : USE HAT1 FORCED_CORNERS
BTN H1U /P MSY(2+) /R MSY(2-)
BTN H1R /P MSX (**2-**) /R MSX (**2+**)
BTN H1D /P MSY (2-) /R MSY (2+)
BTN H1L /P MSX (**2+**) /R MSX (**2-**)

Voyons la conversion des touches fléchées : USE HAT2 AS ARROWKEYS

Est transformé en :

USE HAT2 FORCED_CORNERS
BTN H2U /H UARROW
BTN H2R /H RARROW
BTN H2D /H DARROW
BTN H2L /H LARROW

De même : USE HAT2 AS ARROWKEYS - REVERSE_UD, NOHOLD

Est converti en :

USE HAT2 FORCED_CORNERS
BTN H2**U** **D**ARROW
BTN H2R RARROW
BTN H2**D** **U**ARROW
BTN H2L LARROW

Notons comment les positions hautes et basses sont inter changées, et comment l'insertion du modificateur NOHOLD enlève le /H de l'instruction :

Et enfin, voyons l'attribution comme pave numérique : USE HAT4 AS KEYPAD

Est converti en :

BTN H4U /H KP8
BTN H4R /H KP6
BTN H4D /H KP2
BTN H4L /H KP4
BTN H4UR /H KP9
BTN H4DR /H KP3
BTN H4DL /H KP1
BTN H4UL /H KP7

De même : USE HAT4 AS KEYPAD - REVERSE_UD, NOHOLD

Est converti en :

BTN H4U KP2
BTN H4R KP6
BTN H4D KP8
BTN H4L KP4
BTN H4UR KP3
BTN H4DR KP9
BTN H4DL KP7
BTN H4UL KP1

Notons comment les positions de coin sont ainsi inversées quand nous retournons les directions UD, et comment l'insertion du modificateur NOHOLD enlève le /H des instructions.

De même, c'est vrai pour : REVERSE_LR:

USE HAT4 AS KEYPAD - REVERSE_LR

Qui est converti en :

BTN H4U /H KP8
BTN H4R /H KP4
BTN H4D /H KP2
BTN H4L /H KP6
BTN H4UR /H KP7
BTN H4DR /H KP1
BTN H4DL /H KP3
BTN H4UL /H KP9

Notons encore comment les positions de coin sont ainsi inversées. Quand nous forçons les positions de coin, avec :

USE HAT4 AS KEYPAD - FORCED_CORNERS

le compilateur convertit l'instruction en :

USE HAT4 FORCED_CORNERS
BTN H4U /H KP8
BTN H4R /H KP6
BTN H4D /H KP2
BTN H4L /H KP4

Finalement :

USE HAT4 AS KEYPAD - KP5

Entraîne le compilateur à produire une instruction supplémentaire :

BTN H4M KP5

Notez maintenant qu'il n'y a plus de modificateur /H. Le "KP5" est un caractère non répétitif. C'est à peu près tout à ce propos. Nous pouvons y aller…..si vous n'êtes pas déjà parti !!!

5. Instructions de configuration

5.1 Introduction

A la création de cet ouvrage de référence, nous avons discuté sur les instructions d'un fichier de joystick. Maintenant bien que les instructions puissent apparaître à n'importe quel endroit d'un fichier de joystick, nous avons mis à disposition une zone avant les instructions de programmation, pour les instructions de configuration personnalisée. Nous vous avons déjà présenté certaines des instructions de configuration, telles que les instructions de configuration «USE MDEF». Il est vrai que nous n'avons pas embrouillé les esprits plus tôt en prenant le temps d'expliquer ce que nous entendions par instructions de configuration. Donc, faisons le ici, maintenant que vous devriez être un peu plus à l'aise dans la programmation de votre contrôleur.

Les instructions de configuration s'appliquent à l'ensemble d'un fichier. Il n'y a pas d'instructions programmées à une position spécifique du contrôleur, toutefois certaines peuvent être programmées directement sur des instructions de boutons. Elles disent au compilateur comment configurer le contrôleur pour votre simulateur. Elles incluent des instructions qui disent au compilateur quel fichier de macros utiliser en fonction de sa macro-definition, le nombre de fois que vous voulez qu'un caractère soit généré, quel axe vous désirez désactiver, etc., etc.

Beaucoup de ses instructions de configuration sont décrites ailleurs dans cet ouvrage de référence, à coté des sujets auxquelles elles se rapportent. Je me concentrerai par conséquent ici, sur celles qui n'ont pas été étudié en détails ailleurs.

La syntaxe est toutefois différente pour l' HOTAS Cougar en ce qui concerne les instructions de configuration. Donc voici ici, la règle d'or :

Toutes les instructions commencent soit par "USE", soit par "DISABLE".

Toutes les instructions de programmation logique commencent avec "DEF".

Ceci diffère de la syntaxe Thrustmaster précédente. Oh ! Ne paniquez pas à ce stade à propos du terme "programmation logique" qui a pointé son affreux nez à plusieurs endroits. Ce terme est mentionné dans le but d'être complet et il est approprié aux domaines dans lequel je l'ai mentionné, mais c'est un sujet que nous couvrirons à la fin de cet ouvrage de référence étant donné qu'il s'adresse aux programmeurs purs et durs qui se trouvent parmi vous.

5.2 MDEF – Fichier de DEFinitions de Macros

Déclaration de configuration :

USE MDEF *nom_de_fichier_de_macro*

Cette instruction est seulement nécessaire si un fichier de joystick contient des macros (*ce qui doit être privilégié*). Le *nom_de_fichier_de_macro* dans chaque instruction identique est le nom du fichier de macros avec ou sans son extension (.tmm).

Si par exemple, j'ai un fichier de joystick : « Janes_WW2_Fighters.tmj » et que son fichier de macros est « Janes_WW2_Fighters.tmm », alors les instructions MDEF devraient être :

USE MDEF Janes_WW2_Fighters

Il est important que les deux fichiers se trouvent dans le même répertoire, par défaut celui ci est le dossier «Foxy». Toutefois, il est possible de préparer notre logiciel, afin qu'il n'en ait pas besoin, je pense que c'est un usage pratique à conserver, comme pour les HOTAS TM originaux.

NOTES

1. *Les noms de fichiers longs avec espace sont autorisés pour les noms de fichiers de macros.*

2. *Il importe peu que vous mettiez les extensions à la fin des noms de fichiers dans les instructions* MDEF*. Donc les deux exemples suivants sont corrects :*

USE MDEF Janes WW2 fighters
USE MDEF Janes WW2 fighters.tmm

3. *Les noms de fichiers de macros, noms de fichiers de joystick et noms de macros n'ont pas d'importance :*

Mig Alley.tmm

Est le même fichier de macros que :

mig alley.tmm

4. *Avec les HOTAS TM originaux, vous pouvez actuellement utilisé des instructions* MDEF *multiples, et en conséquence utiliser des définitions de macros depuis différents fichiers de macros. Ceci n'est pas le cas pour le HOTAS Cougar.*

5.3 REPETITION

Indépendamment du reste des instructions de configuration ci-dessous, la majorité des instructions de configuration se rapporte à la programmation des axes, et sont traitées dans cette partie de cet ouvrage de référence.

Déclaration de configuration :

USE RATE (*nnnn*)

Syntaxe de commande

RATE (*nnnn*)

nnnn exprime un temps en millisecondes (1000ms = 1 seconde). Cela détermine le nombre de fois que chaque caractère va être répété. En cas d'oubli, le compilateur mettra par défaut un USE RATE (0) ayant pour résultat de reproduire les caractères à la vitesse de répétition de caractère du clavier par défaut. Plus la valeur de répétition est élevée, plus la vitesse à laquelle les caractères sont répétés est lente. Les valeurs de répétition comprises entre 0 et 655350 (*un peu plus de 10 minutes !)* sont autorisées.

Il est aussi possible de changer la valeur de répétition en temps réel, en la programmant sur un bouton :

BTN S4 /I RATE (100)
/O RATE (0)

ou sur l'instruction d'un axe :

ANT 2 5 RATE (0) RATE (30) RATE (60) RATE (90) RATE (120)

(*voir plus loin les notes sur la programmation des instructions des axes digitaux*)

NOTES AVANCEES

L' HOTAS Cougar reproduit des caractères en groupes appelés "frames". Les "Frames" sont reproduit par intervalle de 30ms environ avec une instruction RATE(0) *(ou sans RATE). A l'intérieur d 'un "frame", le Cougar peut reproduire 16 caractères. Si plus de caractères sont reproduits et que le "frame" contient déjà les 16 caractères maximum autorisés, alors les caractères en surplus seront reportés dans le "frame" suivant. La valeur de répétition (*RATE *value) représente effectivement l'intervalle de temps entre chaque "frames".*

5.4 S3_LOCK et S3_UNLOCK

Déclaration de configuration :

USE S3_LOCK

Syntaxe de commande

S3_LOCK
S3_UNLOCK

Normalement, le BTN S3 sur le joystick est utilisé de telle façon que, lorsque l'on appuie dessus, toutes les instructions /I sont reproduites, et lorsqu'il est relâché, toutes les instructions /O sont reproduites.

Une instruction USE S3_LOCK signifie que si vous appuyez sur le bouton S3, alors le Cougar utilisera seulement les instructions /I. Quand vous appuierez une nouvelle fois sur le bouton, le Cougar basculera sur les instructions /O.

Si vous voulez être capable de basculer entre ses deux états alors vous pourrez utiliser quelque chose de ce genre :

BTN S2 /T S3_LOCK /T S3_UNLOCK

Vous n'avez pas besoin d'avoir une Déclaration de configuration: USE S3_LOCK présente dans un fichier pour utiliser l'instruction directe S3_LOCK, S3_UNLOCK sur un bouton.
Donc quelle est la différence entre USE S3_LOCK et juste S3_LOCK ?
USE S3_LOCK s'applique à la totalité du fichier et est active sitôt que le fichier est téléchargé et activé. Alors que S3_LOCK en instruction de bouton ne s'applique seulement, qu'une fois le bouton appuyé.

NOTES

1. *Si vous attribuez un bouton/chapeau différent de S3 (voir la partie suivante) alors les instructions s'appliqueront à ce bouton, mais vous utilisez la même syntaxe.*

2. *Vous ne pouvez pas utiliser l'ancien modificateur barre oblique* (/H) *avec l'instruction* S3_LOCK. *Donc :*

BTN S1 /H S3_LOCK
BTN S4 /H *Some_macro* S3_LOCK

Produiront tous deux une erreur de compilation.

5.5 Attribution d'un autre bouton pour /I, /O avec SHIFTBTN

Déclaration de configuration :

USE *identifiant_bouton* AS SHIFTBTN

Syntaxe de commande

SHIFTBTN (*identifiant_bouton)*

Exemples :

USE S4 AS SHIFTBTN
BTN T6 SHIFTBTN (T10)

Déterminent quel bouton à utiliser à la place de S3 pour sélectionner l'instruction /I. Si cette instruction est manquante, et sera probablement rarement utilisée, alors le bouton S3 est utilisé pour les instructions /I, /O.

5.6 SENSIBILITE DU «HAT» – USE HAT SENSITIVITY

Il peut être parfois très difficile de déterminer facilement les positions intermédiaires des coins d'un chapeau (hat). (ex. H4UL). Souvenez vous que les chapeaux sont principalement des boutons à 4 positions. Par conséquent, parce que les contrôleurs traitent tout tellement rapidement, vous pourrez souvent d'abords voir l'une des macros produites de l'un ou l'autre cotes de la position de coin, avant de frapper la position de coin. Vous pouvez réduire la sensibilité du chapeau afin de supprimer ce problème avec :

Instructions de configuration

USE *HatID*_SENSITIVITY (*nnnn*)

Où :

HatID designe l'un des 4 chapeaux : HAT1, HAT2, HAT3, HAT4
nnnn est une valeur comprise entre 0 (le plus sensible) et 1000 (le moins sensible). Cette valeur *nnnn* exprime une durée en millisecondes. Par exemple :

USE HAT1_SENSITIVITY (100)

Signifierait que toutes les instructions sur le HAT1 seraient transmises seulement après avoir appuyé 100 ms sur la position du chapeau, laissant plus de temps pour fermer les positions de l'un ou l'autre coté d'une autre position.

NOTES

Vous ne pouvez pas utiliser le modificateur /T *sur l'une des positions programmables du chapeau si vous avez une instruction* USE *HatID*_SENSITIVITY (*nnnn*) *configure pour ce chapeau.*

Donc : USE HAT1_SENSITIVITY (*60*)
BTN H1U /T a /T b /T c produirait une erreur de compilateur.

5.7 UTILISER LA SENSIBILITE DU BOUTON T1

S'il vous arrive sans arrêt d'appuyer sur la touche T1 du micro stick par accident, il vous est possible de réduire sa sensibilité.

Déclaration de configuration:
USE T1_SENSITIVITY (*nnnn*)

Où :

nnnn est une valeur de 0 (le plus sensible) à 1000 (le moins sensible). *nnnn* exprime une durée en millisecondes, et représente le laps de temps avant lequel le bouton T1 appuyé sera reconnu. Cette fonction est intégrée pour les personnes qui pensent qu'elles appuient trop facilement par accident sur le bouton T1.

Voici un exemple :

USE T1_SENSITIVITY (1000)

Signifierait que le bouton T1 ne sera reconnu qu'après une durée de 1 seconde.

NOTES

Vous ne pouvez pas utiliser le modificateur /T *avec une instruction* BTN T1 *si vous avez une instruction* USE T1_SENSITIVITY (*nnnn*) *dans votre fichier.*

5.8 USE FOXY GRAPHIC et README

Déclaration de configuration:

USE FOXY GRAPHIC *imagefile*
USE FOXY README *textfile*

Ces deux instructions sont ignorées par le compilateur et sont seulement utilisées par Foxy pour son propre usage. Quand Foxy ouvre vos fichiers, il les examine, et si il rencontre une instruction USE FOXY GRAPHIC *imagefile*, il chargera ce *fichier image* dans un visionneur d'images. Ceci est pratique lorsque vous échangez vos fichiers, vous obtenez ainsi une représentation graphique montrant quelles macros sont assignées à quels boutons ou chapeaux. Un exemple est :

USE FOXY GRAPHIC Total Air War.bmp

Les fichiers autorisés sont les fichiers bitmaps (.bmp), jpegs (.jpg) ou gifs (.gif).

De même, l'instruction USE FOXY README informe Foxy de charger un fichier texte dans un l'éditeur de modèles, qui peut être utiles aux autres, ou vous servir de pense-bête, en décrivant comment vous avez conçu vos fichiers, quels paramètres sont nécessaires au simulateur pour fonctionner avec, etc. etc. Un exemple est :

USE FOXY README Total Air War.rtf

Les fichiers texte autorisés sont soient de simples fichiers texte, avec l'extension .txt, ou des Rich Texte Files, avec l'extension .rtf. Les RTF peuvent contenir du texte couleur, mis en forme, de fontes diverses, etc., et être alors plus facile à lire.

NOTES

Les fichiers texte et graphique doivent être placés dans le même répertoire que les fichiers macro et joystick. Par défaut, celui-ci est le dossier de fichiers Foxy.

5.9 NULLCHR – Caractère Nul ^

Déclaration de configuration:

USE NULLCHR *character*

Un caractère nul est un caractère dans une instruction qui ne produit aucune sortie. Le caractère nul par défaut est le signe **^**. Donc pourquoi je vous entends prononcer le mot «Problème» ?

Eh bien, en fait, certaine instruction exige un nombre de paramètres fixe pour être valide. Les instructions de type numérique (digital) sont de bons exemples, nous approfondirons le sujet dans la partie suivante. Par exemple, l'instruction :

RDDR 3 L ^ R

programme le palonnier (rudder) pour produire un caractère "L" majuscule quand la pédale gauche est poussée vers l'avant, et un caractère "R" majuscule quand la pédale droite est poussée vers l'avant. Quand le palonnier est au repos et centré, je souhaite qu'il ne se passe rien, donc j'ai ajouté un caractère nul par défaut, "^" pour la position centrale. Je ne pouvais pas laissé l'espace vide comme ci dessous :

RDDR 3 L R

Parce que cette instruction exige 3 caractères après "RDDR 3", autrement le compilateur génèrerait une erreur. Donc pensez au caractère nul en remplacement, là où vous avez besoin d'un caractère, mais que vous souhaitez que rien ne se produise.

Revenons donc à cette instruction, le caractère nul par défaut comme je l'ai dit est le signe (**^**). Si vous désirez utiliser un caractère différent, alors vous pouvez le faire avec une instruction comme :

USE NULLCHR TAB
USE NULLCHR z

Si un signe est utilisé dans un jeu, vous pouvez toujours l'attribuer à des commandes indirectement en entrant SHF 6 dans l'attribution, ex. BTN S1 SHF 6

NOTES

1. *Avec les contrôleurs TM originaux, vous étiez toujours averti de ne jamais laisser une instruction vide s'il se trouvait dans un fichier joystick, et d'ajouter un caractère nul, comme ci-dessous :*

```
BTN S2 /U Fire_Missile
       /M ^
       /D  ^
```

Ceci n'est pas le cas avec le Cougar. Il n'y a aucun problème s'il y a :

```
BTN S2 /U Fire_Missile
       /M
       /D
```

dans votre fichier joystick.

2. *Le caractère nul exécute le code USB* (00). *Cela ne produit rien, il est alors possible si vous voulez configurer une macro dans votre fichier de le faire ainsi :*

Do_Nothing = USB(00)

Et alors d'utiliser ceci dans vos instructions :

RDDR 3 L Do_Nothing R

Bien sur, il est beaucoup plus rapide, facile et simple d'utiliser le signe ^ et c'est en conséquence la raison de son existence.

3. *Vous ne pouvez pas utiliser de combinaisons de touches avec l'instruction* USE NULLCHR*. Car cela produira une erreur de compilateur :*

USE NULLCHR SHF F1
USE NULLCHR ALT p

5.10 CLAVIER - KEYBOARD (AZERTY, QWERTY)

Déclaration de configuration:

USE KEYBOARD *Keyboard type*

Où *Keyboard type* est : soit AZERTY, soit QWERTY.

Si vous utilisez un clavier français AZERTY, et que le jeu pour lequel vous programmez un fichier de remappage clavier afin de correspondre à la disposition du clavier, mais que celui-ci n'exécute pas correctement les fonctions dans le jeu, alors ajouter l'instruction USE KEYBOARD AZERTY dans votre fichier pour voir si cela corrige le problème. Referez vous à la partie Test touche pour plus d'informations.

NOTES

Ne vous ennuyez pas en utilisant l'instruction USE KEYBOARD QWERTY. *Le compilateur créera toujours une erreur en compilant les fichiers. Il n'est pas nécessaire d'avoir cette instruction dans un fichier.*

5.11 Utilisation des profils à partir du panneau de contrôle Cougar - USE PROFILE

Déclaration de configuration:

USE PROFILE *Profile* (Calibration Mode)

Où:

Profile: est un profil créé avec le panneau de contrôle Cougar et sauvegardez avec l'extension .tmc dans le dossier Profiles du logiciel Cougar.
Calibration mode: est soit AUTO, soit CUSTOM. Utilisez ceci pour déterminer si vous préférez que le panneau de contrôle Cougar utilise le profil en autocalibrage ou avec un calibrage que vous aurez configuré.

Exemples:

USE PROFILE Crimson Skies.tmc (AUTO)
USE PROFILE Mechwarrior 4 (CUSTOM)

Comme nous pourrons bientôt le voir avec la programmation des axes, il est possible de modifier les axes, de façon à les intervertir, les désactiver, leurs affecter une courbe de réponse différente, etc. etc. Cela peut être tout à fait compliquer à configurer avec des instructions aussi diverses. Quoiqu'il en soit, si vous utilisez le panneau de contrôle Cougar pour configurer vos profils, et les sauvegarder, alors il sera bien plus facile de les utiliser. Utiliser un profil sauvegardé a aussi l'énorme avantage de pouvoir intégrer les zones mortes (deadzones) dans les courbes de réponse des axes, ce qui ne peut calibré par la programmation. Par ailleurs, par mon expérience, les téléchargements sont plus rapides en utilisant un profil comparé à l'utilisation des instructions DISABLE ou USE AXES_CONFIG (*voir notes plus loin.*)

5.11.1 Informations complémentaires de Profil

Je vais passer un peu de temps ici pour parler des fichiers .tmc, ex. profiles, et pourquoi c'est une idée très intéressante de les inclure dans tous vos fichiers joystick. Tout d'abords, les profils sont créés en utilisant le Panneau de Contrôle Cougar (CCP). Ils contiennent toutes les informations que vous pouvez configurer dans le CCP se rapportant aux axes, comme le mappage des axes, zones mortes (deadzones), courbes de réponses, etc.
Maintenant si vous téléchargez un fichier qui simule n'importe quelles données d'un axe, ou si vous utilisez un fichier qui vous permet d'effectuer un changement de vos courbes tout en volant, alors quand vous quitterez le simulateur, toutes ces informations resteront enregistrées dans le contrôleur. Souvenez-vous, ceci est un système sans pilotes et toutes les informations sont enregistrées dans le

contrôleur. Si vous téléchargez et utilisez un fichier différent pour un autre simulateur, cela ne remettra pas à zéro les informations des axes, alors vous conserverez les valeurs du simulateur précédent, parce qu'elles seront toujours enregistrées dans le Cougar. Vous avez ainsi 2 possibilités pour contourner cela si ça vous pose problème. Vous pouvez soit utiliser USE PROFILE dans chacun de vos fichiers joystick, en le configurant dans le fichier DEFAULT.tmc (*le profil par défaut est créé par le CCP lors de sa première utilisation*) ou un profil de votre choix pour ce sim, ou avec Foxy, vous pouvez à partir du menu de téléchargement, demander au compilateur, quand il charge un fichier, de commencer par faire un reset des axes en appliquant le fichier choisi, et à chaque chargement du fichier. J'espère que cela a un sens.
Pour finir, parce qu'un fichier contient des données de calibrage, ceci est la raison pour laquelle vous devez dire au compilateur si vous voulez utiliser les données de calibrage dans votre sim, ou si vous préférez activer le calibrage Auto dans un profil.

NOTES

1. *Foxy et le compilateur supposeront que les profils sont dans le dossier Profiles du logiciel Cougar. C'est ici que tous les profils créés avec CCPsont sauvegardés. Par défaut:* C:\Program Files\Hotas\Profiles. *Lorsqu'un fichier est compilé/chargé en contenant* USE PROFILE*, alors le compilateur :*

- *(a) Apparaîtra pour le profil dans le répertoire Profiles par défaut.*
- *(b) Si le profil existe, il utilisera celui ci (et ne cherchera pas ailleurs).*
- *(c) Si le profil n'existe pas, il sera dans le même répertoire que le fichier joystick file, dossier Foxy.*
- *(d) Si le fichier n'existe toujours pas, cela produira une erreur.*

2. *Les profils ont l'extension* **tmc**. *Ca n'a pas d'importance si vous avez cette extension avec l'instruction* USE PROFILE. *Donc:*

USE PROFILE Crimson Skies.tmc *est équivalent à*
USE PROFILE Crimson Skies

3. *Si vous utilisez Foxy pour ouvrir un fichier d'une autre personne, et que ce zip contient un profil, alors tous les fichiers vont être dezippé dans le dossier Foxy. Vous pouvez choisir si vous voulez déplacer les profils dans le dossier de profils du logiciel Cougar bien que Foxy ne le fera pas pour vous. Vous devrez utiliser l'Explorer. Je conseille de laisser tous les profils dans le dossier HOTAS Profiles* (*normalement* C:\Program Files\HOTAS\Profiles).

4. *Un profil contient principalement les infos suivantes à propos des axes Cougar: Donnés Mappage, directions des axes, positions centrales, données de calibration, Zones mortes, Courbes, Trim, données d'activation d'axes et de visualisation.*

5.12 Instructions de configuration décrites ailleurs dans cet ouvrage de référence

Certaines instructions sont au delà de la portée de ce chapitre cependant elles nécessitent une explication en regard des autres instructions de configuration. Les instructions suivantes sont expliquées ailleurs :

Instructions de configuration	Description
USE *Btn* AS DXn	Section 3.8: Définit les boutons et fonctions avec DirectX (Direct Input)
USE ALL_DIRECTX_BUTTONS	Section 3.8.1: Attribue tous les boutons en tant que boutons DirectX
USE HAT AS MOUSE, POV, ARROWKEYS, KEYPAD	Section 4.1: Programmation des chapeaux du Joystick
USE CURVE	Section 6.3: Courbes de réponse et (CURVE)
DISABLE AXIS	Section 6.5: Désactivation des axes
USE SWAP	Section 6.6: Mappage des axes (SWAP)
USE REVERSE	Section 6.7: Inverse la direction d'un axis
USE AXES_CONFIG	Section 6.8: Instruction USE AXES_CONFIG
USE MTYPE	Section 7.2: USE MTYPE –la façon la plus simple pour attribuer la souris sur le micro stick
USE Axis_Identifier AS Mouse_Axis	Section 7.3.1: Attribution d'autres axes sur les axes de la souris
USE ZERO_MOUSE	Section 7.5: Evite le blocage des mouvements de souris avec des instructions souris personnalisées et /I, /O
DISABLE MOUSE	Section 7.7: Désactive l'attribution par défaut de la souris sur le micro stick
USE SCREEN_RESOLUTION	Section 7.8.1: Définit la résolution d'écran
DEF Xn	Section 8.2: Définition des repères logiques et des instructions bouton

6. Programmation des axes

6.1 Principes de base

6.1.1 Différences entre analogique et numérique

Nous allons maintenant examiner comment programmer les différents axes Cougar de façon numérique, et comment modifier leurs comportements analogiques. D'abords, il est nécessaire d'expliquer la différence entre un axe numérique et analogique, beaucoup trouvant ces termes déroutants.

La plupart de joysticks sur le marché actuellement fonctionne de façon identique : mécaniquement. A l'intérieur, se trouvent deux potentiomètres, ou **pots** auxquels ils se rapportent. Si vous avez une radio Hi-Fi qui possède une molette que vous tournez pour régler le volume, ce que vous tournez alors est un pot qui varie sa résistance quand on le tourne.

Dans un joystick, ces pots sont disposés perpendiculairement l'un à l'autre, pour mesurer le déplacement gauche/droite du joystick, l'axe x, et le déplacement avant/arrière, l'axe y. Donc un joystick possède deux axes principaux le long desquels il se déplace, les axes x et y. La position du joystick lorsque vous le bougez peut être déterminer par ces deux axes, ex. A quelle distance le manche s'est déplacé le long de l'axe x, et à quelle distance le long de l'axe y. Les pots donnent aussi une série de valeurs lorsque vous déplacez votre joystick, et en tant que tel, ils sont appelés appareils analogiques, en opposition au terme numériques, et qui sont soient sur marche soient sur arrêt, comme les touches de votre clavier, ou les boutons de votre joystick.

Maintenant, si vous êtes toujours là et que vous n'êtes pas passé au chapitre suivant, alors compliquons un peu ici la question. Dans un monde parfait, les pots vous donneraient des valeurs très précises et tables, dans toutes positions. Mais les pots peuvent subir des dommages causés par l'usure et la poussière. L'effet est de produire parfois de légères variations de valeurs, ou pire, des "sauts" (ou "pointes") dans les valeurs. Avec le Cougar, les signaux provenant des pots ne sont pas envoyés directement vers votre sim. Un processeur numérique à l'intérieur du Cougar lit les valeurs provenant d'un pot, et sépare alors les fausses valeurs, pour fournir une valeur plus précise et plus stable. Donc bien que les pots sont *analogiques*, les *signaux* provenant de ceux-ci sont *numériques*.

Nous examinerons bientôt ce qu'il est possible de faire avec les axes du Cougar, et une chose que l'on peut faire est de le programmer numériquement. Avec ce qui précède, il est nécessaire d'expliquer plus loin, parce que ce n'est pas évident de voir comment programmer un axe analogique de façon numérique. Disons à titre d'exemple, que la manette produit des valeurs de 0 à 100. Il est alors

possible de diviser les axes en 5 plages, où la plage 1 équivaut à une lecture de 0 à 19, la plage 2 à une lecture de 20 à 39 etc. Et nous pouvons imaginer une instruction qui dit "Quand nous sommes dans la plage 1 produire un 'a', en plage 2 un 'b' etc. Nous désignons cela en programmant un axe avec une instruction de type digital, que j'expliquerai plus loin dans le chapitre suivant.

6.1.2 Les axes du Cougar

Le Cougar possède 10 axes analogiques. Ces axes sont considérés comme :

- Purement analogiques, *(assurant que vos jeux et DirectX les supportent*) et peuvent par conséquent leurs attribuer des fonctions
- Purement digitaux, afin de les programmer pour produire juste des caractères clavier
- Ou une combinaison des deux.

De plus, nous pouvons assigner un axe analogique en :

- Les déplaçant complètement
- Leurs appliquant des courbes de réponse différentes
- Leurs appliquant des valeurs trim.
- Inversant la direction des axes
- Les mappant sur d'autres axes, tous deux par défaut comme basé sur la position des switchs dogfight et S3

L'une des caractéristiques du Cougar est le nombre de possibilités de ses axes. Quoiqu'il en soit cela peut être très rapidement compliqué! Donc pour essayer de simplifier notre compréhension sur ce que nous pouvons accomplir dans un fichier en terme de programmation de ces axes, nous devons d'abords définir les **6 Instructions Digitales**. Celles ci sont utilisées pour programmer les axes numériquement pour produire les caractères clavier. Apres nous serons en bonne position pour voir ce que nous pouvons faire avec les axes analogiques pour que le sim les voit. Avant d'aller plus loin, définissons la syntaxe des 10 axes:

Syntaxe TM	Axes
JOYX	Joystick X
JOYY	Joystick Y
THR	Throttle (manette des gaz)
RNG	Range (Portée)
ANT	Antenna
MIX	Microstick X *
MIY	Microstick Y
LBRK	Left Toe Brake (Frein pied gauche)
RBRK	Right Toe Brake (Frein pied droit)
RDDR	Rudder (Palonnier)

* Note : Les Microstick X, Y sont différents de Mouse X, Y

NOTES

1. *Avec les HOTAS TM originaux, un axe était soit analogique ou numérique, il était reconnu par défaut dans le jeu et une fonction lui était attribuée (ex. TQS Throttle = Puissance, gaz) ou programmé numériquement pour produire les caractères clavier. Ce n'est pas le cas avec le nouvel HOTAS Cougar. Par défaut, les axes sont reconnus comme analogique mais si vous les programmez numériquement, alors ils seront à la fois analogiques* ***et*** *digitaux. Si un axe purement numérique est nécessaire alors l'axe en question devra être désactivé.*

2. *Vous n'avez pas besoin de changer quoique ce soit dans le CCP, Application Options Jeux si vous voulez seulement utiliser un axe numériquement. C'était le cas avec les HOTAS TM précédents, Mais pas avec le Cougar.*

3. *Avec toutes les instructions d'axes digitaux :*

- *Vous pouvez utiliser les instructions /U, /M, /D, /I, /O*
- *Toute modification d'axe analogique (course, plan) etc. ne modifiera pas les instructions digitales. Ils restent sur l'axe physique, et linéaire.*

6.2 Instructions de Type Digital

Dans ce chapitre, nous verrons comment programmer les axes de façon numérique pour produire des caractères clavier. Programmer un des axes pour produire des caractères clavier est possible en utilisant l'une des 6 instructions de type digitale disponibles. La manière la plus simple de comprendre ces 6 instructions est de voir un exemple de chaque et le caractère ainsi produit.

Note: *Ne voulant pas compliquer les choses de suite, mais juste pour garder au fond de votre esprit que ce ne sont pas de simple caractère qui doivent être utiliser avec ce type d'instructions. Les signaux logiques, instructions souris et instructions de courbes d'axes peuvent aussi être utilisé.*

6.2.1 Type 1: Répétition de caractères

Une instruction de type 1 a la syntaxe suivante :

Identifiant d'axe	Instruction de Type Digital	Nombre de caractères ou macros (max. 50)	Caractère ou macro haute	Caractère ou macro basse	caractère ou macro central *(optionnel)*	FORCE_ MACROS *(optionnel)*
Ex. ANT	1	10	u	d	c	- FORCE_MACROS

Ou comme il devrait apparaître dans un fichier joystick :

ANT 1 10 u d c - FORCE_MACROS

En tournant ANT dans le sens des aiguilles d'une montre puis dans le sens opposé devrait produire les caractères suivants :

u u u u u c u u u u u d d d d d c d d d d d

Nous ne sommes pas limité à l'utilisation de caractères simples dans les instructions digitales, donc nous pourrons avoir des macros définies comme suit :

Chaff_Flare = c DLY(30) f
Getting desperate = RPT(20) (c f)

Et avoir une instruction de Type pour le bouton RNG dans le fichier joystick :

RNG 1 5 Chaff_Flare Getting desperate

Notez que le Nombre de caractères définit maintenant le nombre total de caractères produits (*incluent les caractères de centre*) pour la course **totale** de l'axe. C'est la différence d'avec les instructions TM d'origine de Type 1, où le nombre de caractères définissait le nombre de caractères produits de début de course jusqu'au caractère central, puis du caractère central jusqu'en fin de course. La raison pour ce changement de syntaxe est du au fait que maintenant nous mettons le caractère central optionnel dans l'instruction. Par exemple :

RNG 1 6 u d

Produit les caractères suivants en tournant le bouton RNG :

u u u u u u d d d d d d

Notez que l'insertion d'un caractère central **est différent** de l'utilisation du caractère nul (qui est par défaut ^) pour le caractère central. Donc l'instruction :

RNG 1 6 u d ^

Produit :

u u u *dead-zone* u u u d d d *dead-zone* d d d

Le caractère Nul "^" n'a donc aucun résultat… une sorte de zones mortes si vous voulez. Avant de quitter ce sujet, le Nombre de caractères **doit être un nombre pair** *si* un caractère central est fourni. Autrement s'il n'y a aucun caractère de centre, ça peut être n'importe quoi. La raison de cela, je l'espère, est évidente : si le nombre de caractère requis est 20 et qu'il y a un caractère de centre, alors vous devez être capable d'en avoir 10 de chaque coté du caractère de centre.

6.2.1.1 Comprendre le modificateur FORCE_MACROS

Ce modificateur est optionnel, et ne peut être utilisé qu'avec les instructions digitales de Type 1, 2, 5 et 6. Je vais utiliser une instruction de Type 1 pour expliquer la signification de ceci, mais ce qui suit s'applique également aux autres types d'instructions avec lesquelles il peut être utilisé.

Disons que nous avons ceci :

RNG 1 50 u d

En tournant le bouton RNG d'un bout de course à l'autre, ce la produira 50 caractères "u", ou 50 "d", cela dépends du sens dans lequel on le tourne. Maintenant si on le tourne rapidement, en testant le résultat dans le bloc notes ou le testeur Foxy, nous ne verrons pas 50 caractères. Vous en verrez 10 20 mais pas 50. Donc que se passe-t-il ? Est ce un bug ? Est ce que je veux dire que le Cougar est fait pour être capable d'exécuter les instructions très rapidement ? Oui, le Cougar exécute les instructions très rapidement, et en parallèle, et c'est actuellement la raison pour laquelle vous observe cet effet. Que ce passe-t-il quand vous tournez le bouton RNG rapidement ? Est ce parce que certains des caractères sont manquants. C'est simplement que les processus d'appuie et de relâchement des touches sont exécutés en parallèle et que si plus de 16 caractères ont été exécuté, l'ordinateur les verra juste comme un caractère dans une même «frame». Nous pouvons changer ce comportement en forçant l'ordinateur à voir chaque caractère, de la façon suivante :

RNG 1 50 (< u >) (< d >)

Et c'est la raison d'être du modificateur FORCE_MACROS. Il entoure chaque caractère/macro dans une instruction digital avec (**<** *character/macro* **>**). Faites tout de même attention à la façon dont vous utiliser ce modificateur. Si vous forcez l'instruction d'une position alors les autres instructions ne seront pas produites tant que l'instruction forcée sera exécutée. De ce fait, assurez vous qu'aucune macro utilisée dans une instruction digitale FORCE_MACROS n'ait le modificateur forcé (les crochets < >) dans sa définition. Vous ne pouvez pas superposer des attributs forcés et vous ferez ainsi si vous avez :

Macro_1 = < a b c>
Macro_2 = d

RNG 1 50 Macro_1 Macro_2 - FORCE_MACROS

Comme le compilateur devrait le convertir en :

RNG 1 50 (< < a b c> >) (< d >)

NOTES

1. *Vous ne pouvez avoir :* RNG 1 3 a b c d e f

mais vous pouvez avoir RNG 1 3 (a b c) (d e f)
ou RNG 1 3 ABC_macro DEF_macro

et dans le fichier macro: ABC_macro = a b c
DEF_macro = d e f

2. *Vous pouvez utiliser* /U, /M, /D, /I, /O *avec toutes les instructions digitales, ex :*

RNG /U 1 3 F1 F2
/M 1 5 (SHF UARROW) (SHF DARROW)
/D /I 1 6 e t F5
/O 1 4 [ ] KP5

3. *Vous pouvez aussi utiliser* /P, /R *et* /H *avec vos macros ou avec crochets directement dans vos instructions digitales de type 1, 2, 5 et 6. Par exemple :*

RNG 1 3 Macro_1 Macro_2

Et dans votre fichier macro :

Macro_1 = /P a /R b
Macro_2 = /H d Rem Mais pourquoi faire cela, ça je ne sais pas !

4. *Vous ne pouvez utiliser* /T *avec une instruction Type 1 (ou* ***un*** *axe digital).*

5. *Une instruction Type 6 est une instruction de Type 1 spéciale. Ces instructions produisent une réponse identique depuis le bouton ANT* (u u c u u d d c d d) :

ANT 1 4 u d c
ANT 6 5 (0 20 40 60 80 100) u d c

6.2.1.2 Considérations important l'utilisation de FORCE_MACROS

NOTES AVANCEES

Il apparaîtrait que c'est une bonne idée d'utiliser FORCE_MACROS *à longueur de temps d'après l'explication ci-dessus, mais c'est faux pour plusieurs raisons. La première chose à penser est l'effet pas seulement sur la réponse des autres boutons programmés du contrôleur mais aussi sur la réponse des axes pour lesquels vous avez ajouté* FORCE_MACROS. *Examinons un exemple, qui me permettra aussi d'expliquer autre chose.*

Considérons l'instruction de l'axe digital de Type 2 (oui, je sais que je ne l'ai pas encore traité. Je le fais dans la partie suivante et ceci est assez facile à comprendre.)

ANT 2 26 a b c d e f g h i j k l m n o p q r s t u v w x y z

Maintenant pas besoin d'être un génie pour comprendre que, ignorant la syntaxe, l'effet de ces instructions est de programmer le bouton ANT sur la manette des gaz afin de produire les caractères de l'alphabet.

(Toutes mes excuses si votre alphabet est différent du mien ! Et pour ce qui est de mon public américain, nous prononçons 'z', 'zed' de ce coté de l'Atlantique et non pas 'zee', mais cela n'a aucune importance. Je n'ai d'ailleurs aucune idée du pourquoi je mentionne cela ici. Je recommence à radoter.)
Maintenant si vous ajoutez cela à votre fichier joystick, et que vous le téléchargez, alors quand vous allez tourner le bouton ANT, cela va produire les caractères de l'alphabet, et vous pouvez utiliser le testeur de touches de Foxy pour les voir. Vous remarquerez que les caractères sont produits de façon très rapide. Si vous changez alors les instructions en ajoutant le modificateur FORCE_MACROS*, de cette façon :*

ANT 2 26 a b c d e f g h i j k l m n o p q r s t u v w x y z - FORCE_MACROS

Et que vous faites la même opération, vous remarquerez que le débit du bouton ANT est beaucoup plus lent. En effet, il est au moins 4 fois plus lent. Ceci est du au fait que pour chaque caractère, par exemple le "a", le compilateur le convertit en : < KD(a) KU(a) >. *Chacun de ces 4 éléments ira dans son propre "frame".*

Je ferais mieux d'expliquer le terme «frames» avant de continuer. Quand le Cougar veut exécuter des caractères ou des instructions programmées, il les envoie toutes les 30ms. Si vous voulez que cela soit le taux de «frame» du Cougar. Chaque frame peut contenir plusieurs caractères. Il n'y a pas qu'un caractère par frame (souvenez-vous que le Cougar traite en parallèle, plus de 32 macros en même temps). Donc dans le premier exemple que j'ai donné pour ANT et l'alphabet, l'une des raisons pour lesquelles il produit l'alphabet aussi rapidement est du au fait qu'il envoie plusieurs de ces caractères en une "frame" dépendant de la vitesse à laquelle vous tournez le bouton ANT. Vous voyez actuellement cela dans le testeur de Foxy. Beaucoup des évènements Key Down *se produisent ensemble, suivi dans le frame suivant par leurs évènements* Key up. *Je reviendrai sur tout ça…*

De toute façon, revenons au modificateur FORCE_MACROS*, maintenant que le "a" a été converti en* < KD(a) KU(a) >. *Ces 4 composants de l'instruction* (<, KD(a), KU(a) et >) *vont aller dans des frames séparés, et alors l'alphabet complet se produit **plus** lentement.*

Pendant que nous parlons de l'instruction ANT*, je vais m'éloigner un peu du sujet en expliquant une autre caractéristique du Cougar. Cette capacité d'envoyer des caractères multiples dans le même frame, et l'un des effets qui se rapporte à l'ordre dans lequel les caractères sont produits. Examinons encore l'instruction :*

ANT 2 26 a b c d e f g h i j k l m n o p q r s t u v w x y z

A en croire ces instructions, si on tourne ANT *dans un sens, cela va produire les caractères dans l'ordre alphabétique, et si on tourne dans l'autre sens, l'opposé. Regardons ce qui se passe quand on tourne rapidement le bouton. Vous verrez dans le testeur Foxy que certains caractères apparaissent dans le désordre. Ca se remarque plus dans le sens inverse des aiguilles d'une montre, ce qui produit :*

"x y z u v w p q r s t m n o j k l f g h i a b c d e"

Vous verrez aussi dans les codes clavier Windows, que les caractères sont produits comme un groupe de minuscules, suivi par un groupe de majuscules, ex. Down (z y x) Up (z y x) *à la place de* Down (z) Up (z) Down (y) Up (y) *etc. Donc pourquoi voyons nous ce désordre en tournant* ANT *dans ce sens? Est ce un bug? Non. Cette la façon dont l'USB envoie les codes clavier. Avec les anciens contrôleurs TM, les caractères étaient envoyés avec le standard PS2 qui dissocier les minuscules et majuscules. Avec l'USB, ça fonctionne comme suit : les caractères sont envoyés comme si, le système examinait un buffer clavier et qu'il notait quelle touche a été appuyé ou non. Donc quand le Cougar modifie un buffer pour dire que les touches z, x et y sont appuyées, l'OS prends cela comme ça et envoie les caractères dans un ordre défini, qui pour le caractère est l'ordre alphabétique. Donc dans le test, bien que vous ayez demandé au bouton* ANT *d'envoyer les caractères z, x et y dans cet ordre, si* ANT *est tourné assez rapidement alors le Cougar place* z, x *et* y *dans le même frame, et l'ordinateur voit ses 3 caractères comme appuyé en même temps et les affiche dans l'ordre alphabétique, ex.* x, y *et enfin* z.

Bien sur quand on ajoute FORCE_MACROS*, alors nous obtenons toujours l'alphabet dans le bon ordre pour le sens de rotation du bouton* ANT*, parce qu'aucun caractère ne sera placé dans le même frame que les autres caractères produits par le bouton* ANT*. Mais nous les produisons plus lentement.*

J'espère que vous comprenez cela. Donc pensez à ça quand vous utiliserez FORCE_MACROS *dans votre simulateur ou non. Rappelez vous la règle d'or. Ce n'est pas la façon dont il se comporte dans Windows qui importe. Vous devez essayer cela dans vos sims et utiliser celle qui convient le mieux. Une dernière note : Si vous essayez cette instruction* (sans FORCE_MACROS) *alors vous remarquerez que certains caractères ne sont pas produits quand le bouton* ANT *est tourné rapidement. En effet, vous obtiendrez quelques caractères comme* SHF *et* CTL*. C'est une chose que nous connaissons mais qui ne semble pas être un bug avec le Cougar. Nous pensons que c'est en rapport avec les drivers clavier Microsoft finissant par être surchargé au point de se réinitialiser eux mêmes, créant ces erreurs de caractères. Il est impossible pour le Cougar de produire ces caractères faux avec cette instruction.*

6.2.2 Type 2 : Séquence normale de caractères, régions définies

Une déclaration de TYPE2 a la syntaxe suivante:

Identifiant d Axes	Déclaration digitale Type	Nombre de caractères ou macros (max.50)	Séquence de caractères et/ou de macros et/ou de « logical flags «	FORCE_ MACROS *En option*
eg. ANT	2	5	a b c d e	- FORCE_MACROS

Ou tel qu il devrait apparaître dans le "joystick file":

ANT 2 5 a b c d e - FORCE_MACROS

Faire tourner le "ANT KNOB" dans le sens horaire puis dans le sens ante horaire, devrait générer la chaîne de caractères suivantes:

b c d e d c b a

De même qu avec la déclaration de "TYPE 1", notez que chaque caractères n est seulement génère qu une seule fois (mais que le *« logical **flags** »reste sur ON (voir plus description plus loin*). De même cela reste vrai si les macros sont utilisées a la place de caractères seuls. Remarquez également qu avec le « ANT KNOB » débutant un rotation de sa position centrale dans le sens horaire, générera un **b** en premier caractère et non un **a** comme nous aurions pus le penser, c est une action différente compare a une déclaration de « TYPE 1 ». Cela veut dire que si vous continuez a tourner plusieurs fois le « ANT KNOB » sur toute sa course possible, vous aurez en sortie un cycle de caractères comme suit :

b c d e d c b a b c d e d c b a b *etc. etc.*

Et vous ne produirez pas une chaîne comme :

a b c d **e e** d c b **a a** b c d **e e** d c b a *etc. etc.*

Notez que différemment de la syntaxe originale de TM, le nombre de caractères ou de macros pourra être pair ou impair. Manifestement si vous assignez une déclaration a un support comme un "ANT KNOB" vous préférerez probablement utiliser un nombre impair, auquel la position centrale correspondra à la position centrale du « KNOB ».

Vous pouvez également utiliser des macros avec une déclaration digitale de type 2 de manière similaire:

RNG 2 5 Emcon-1 Emcon-2 Emcon-3 Emcon-4 Emcon-5

Et dans le fichier macro :

Emcon-1 = e DLY(40) 1
Emcon-2 = e DLY(40) 2
Emcon-3 = e DLY(40) 3
Emcon-4 = e DLY(40) 4
Emcon-5 = e DLY(40) 5

Maintenant je sais que nous n avons pas encore discute de ce que sont les " logical flags " (drapeaux logiques), mais si vous voulez vous engager dans ce type de programmation sachez juste que vous pouvez directement les utiliser dans une déclaration de « TYPE 2 » incluant des « boutons de logical flags » :

RNG 2 4 X1 X2* X3* X4

Nous verrons cela plus en profondeur dans la section sur « la programmation logique ». Vous pouvez tout aussi bien mélanger, des caractères simples, des macros, et des « logical flags » dans une déclaration de « TYPE 2 « , comme le montre l exemple suivant :

ANT 2 5 a Emcon-2 c ^ X1

6.2.2.1 Comprendre le modificateur - FORCE_MACROS

Voir la section 6.2.1.1 for sur les déclaration de "TYPE 1" .

NOTES

1. *Vous pouvez utiliser les attributs /U, /M, /D, /I, /O "slash modifier" avec toutes les déclarations digitales. De toutes manières, soyez prudent si vous mélanger ces attributs sans aucunes assignations dans une déclaration logique (voir explications plus loin)*

Exemple pour les utilisateurs avertis

```
ANT /U 1 6 a b c
    /M 1 6 d e f
    /D 2 3 (DLY(5000) X1) X2 X3
BTN X1 /U a
       /M b
       /D c
```

Sur la position /D, X1 *peut générer un "a", "b"ou "c" cela dépend si le « dogfight switch » a change de position durant les 5 secondes de délai.*

2. *Vous ne pouvez pas utiliser /U, /M, /D, /I, /O* ***sans déclaration*** *(cela s applique a toutes les déclarations de type digitales. Cela produirai une erreur de compilation.*

RNG 2 3 a b macro_1

where macro_1 = /I KP1 /O KP2

3. *Imaginons que vous avez programme ceci :*

ANT /I 2 3 SM1 SM2 SM3
/O 2 3 SM4 SM5 SM6

where: SM1 = a
SM2 = /H b
SM3 = c
SM4 = d
SM5 = /H e
SM6 = f

Imaginons de même que le "switch S3 "soit en position ***out****, et que l " antenne knob" soit en position centrale (générant «e" en continu), et maintenant pressons le « S3 switch « .il en résultera un arrêt de la génération du « e », le caractère « b » sera automatiquement généré aussi longtemps que S3 sera « pressé. Et lors du relâchement de S3 un « e » continu sera généré de nouveau.*

4. *Vous pouvez utiliser /P, /R, /H avec des déclarations de Type 1, 2, 5 et 6.*

6.2.3 Type 3: Génération continue de caractères

Une déclaration de TYPE 3 a la syntaxe suivante:

Identifiant d Axes	Déclaration digitale Type	caractère gauche	caractère central	caractère droit
Eg. RDDR	3	l	c	r

Ou tel qu il devrait apparaître dans le "joystick file" :

RDDR 3 l c r

Pousser la pédale gauche du palonnier produira un "l" continu, exactement de la même manière que sont généré les caractères continus avec l attribut /H .

NOTES

1. *L axe n est pas divisé en 3 parties égales mais plus comme monte ci dessous, car autrement la zone centrale serait trop importante :*

Région gauche	Région centrale	Région droite
/H l	c	/H r

2. *Vous pouvez utiliser des " logical flag" avec une déclaration de TYPE 3*

3. Si vous ne voulez pas utiliser de caractère central, utilisez un caractère nul (^):

RDDR 3 l ^ r

6.2.4 Type 4: Génération repetee de caractères

Une déclaration de TYPE 4 a la syntaxe suivante:

Identifiant d Axes	Déclaration digitale Type	Taux de répétition (ms)	caractère gauche ou macro	caractère central ou macro	caractère droit ou macro
eg. RNG	4	1000	l	c	r

Ou tel qu il devrait apparaître dans le "joystick file" :

RNG 4 1000 l c r

Un caractère répété est un caractère qui est génère toutes les x millisecondes, un peu comme un phare ou un stroboscope. Les déclarations de Type 4 sont nouvelles pour les contrôleurs TM. Avec la déclaration du "RANGE" ci-dessous, quand le "Range Knob" est tourne vers la gauche (sens horaire si on le regarde de face) alors un caractère "l" sera produit toutes les 1000 millisecondes (1 toutes les secondes).Inversement, le tourner dans la direction opposée générera un seul caractère "c" au passage de a zone centrale puis un caractère "r" toutes les secondes.

NOTES

1. *Macros et « logical flags »peuvent être aussi utilises a la place d un simple caractère.*

2. Si vous ne voulez pas utiliser de caractère central, utilisez un caractère nul (^):
RNG 4 60 l ^ r

3. Le taux de répétition est donne en millisecondes entre *0 et 82800000 (23 heures !)*

6.2.5 Type 5: Sequences programmees de caractères , Zones variables

Une déclaration de TYPE 5 a la syntaxe suivante:

Identifiant d Axes	Déclaration digitale Type	Nombre de zones (max. 50)	Taille des zones (en pourcentage)	Séquence de caractères et/ou macros et/ou « logical flags »	FORCE_ MACROS *(option)*
eg. THR	5	4	(0 20 45 70 100)	a b c d	- FORCE_MACROS

Ou tel qu il devrait apparaître dans le "joystick file" :

THR 5 4 (0 20 45 70 100) a b c d - FORCE_MACROS

A première vue une déclaration de caractères de TYPE 5 parait un peu plus complexe que les autres types, mais dans l essentiel il s'agit juste d un cas dérivé de déclaration de TYPE 2.
Rappelez vous qu une déclaration de TYPE 2 possède une génération de caractères repartie également sur toute la course de déplacement de l axe.
Une déclaration de TYPE 5 partage l axe en régions ou zones, et leur assigne a chacun leur séquence propre de caractères.
Dans l exemple ci-dessous 4 zones sont définies :

- 0 to 20% de déplacement d axes produit un caractère "a"
- 21 to 45%: un caractère "b"
- 46% to 70%: un caractère "c"
- 71% to 100%: un caractère "d"

Dans tout les autres aspects dune déclaration de TYPE 5 , les restrictions et les règles des déclaration s de TYPE 2 sont en vigueur. Maintenant une déclaration « digitale » peut exister sur un axe analogique. Nous pouvons donc avoir par défaut une manette des gaz « analogique » mais utilisant une déclaration de TYPE 5 , générant des caractères sur n importe quels points de déplacement de cet axe. Donc il vous sera très facile d introduire une position « reverse »ou d activer le freinage pendant un atterrissage avec les « gaz au minimum », correspondant a la position la plus basse de la course de la manette de gaz déclarée.

```
THR /U
    /M
    /D 5 1 (0 5) Wheelbrakes
```

et dans le fichier Macro :

```
Wheelbrakes = /P b /R b
```

Donc quand le bouton « Dogfight » est en position basse (/D) et que la « *throttle » est en position minimum, la macro* Wheelbrakes engagera le freinage. Notez que

je ne déclare aucune fonction digitale lorsque le bouton « Dogfight » se trouve en position centrale (/M) et haute (/U) . De ce fait la macro Wheelbrakes est seulement exécuter lorsque le bouton « Dogfight » est en position basse (/D) .

6.2.5.1 Comprendre l attribut FORCE_MACROS

Reportez vous section 6.2.1.1 pour les explications sur les déclarations de TYPE 1 concernant cet attribut.

NOTES

1. Avec les Hotas TM originaux (Flcs, F22 Tqs…) vous ne pouviez pas programmer la position minimale de al manette des gazs avec une déclaration BTN MT*t mais seulement quand le Throttle n était pas analogique. L attribut* BTN MT n est plus supporte, du fait de l augmentation des possibilités que vous avez lorsque vous programmez en digital par rapport a celles disponibles en mode analogique .Si vous voulez émuler une déclaration BTN MT : Prenons pour exemple la déclaration ci-dessous *ex:* THR 5 1 (0 5) Your_macro_here

2. *Comme avec toutes les déclaration s digitales, aucunes « Curves » appliquées a un axe analogique n affectera les déclaration s digitales .Elles gardent leurs propres "Cureves" linéaires.*

3. Vous pouvez utilisez les attributs /P, /R, /H avec des déclarations de *Type 1, 2, 5 et 6 mais placez les dans une macro ou si vous les utilisez directement dans une déclaration , placez les entre parenthèses.*

THR 5 1 (0 5) Wheelbrakes

La ou vous avez declare dans votre macro "Weelbrakes comme :
Wheelbrakes = /P b /R b

Est correcte, mais vous pourrez aussi avoir ::

THR 5 1 (0 5) (/P b /R b) *mais::*
THR 5 1 (0 5) /P b /R b *générera une erreur de compilation.*

6.2.6 Type 6: Génération répétée de caractères , zones variables

Une déclaration de TYPE 6 a la syntaxe suivante::

Identifiant d axe	déclaration digitale Type	Nombre de régions (max. 50)	Taille des zones (%)	haut	bas	Centre *(option)*	FORCE_ MACROS *(option)*
eg. ANT	6	5	(8 20 40 45 70 80)	u	d	c	- FORCE_MACROS

Ou tel qu il devrait apparaître dans le “joystick file” :

ANT 6 5 (8 20 40 45 70 80) u d c - FORCE_MACROS

Une déclaration digitale de TYPE 6 est essentiellement la même qu une délacération de Type 1, a par que les caractères ne sont pas repartis sur des "bandes" de tailles égales , comme cela l est dans une déclaration de TYPE 1 , Au lieu de cela ils sont places sur des "bandes" de tailles choisies.

Contrairement au TYPE 1 , si vous incorporez un caractère centrale alors le nombres de zones totale doit etre **impaire**.

Dans l exemple ci-dessous, 5 zones sont définies:

- 8 to 20% du déplacement de l axe
- 21 to 40%
- 41 to 45%
- 46% to 70%
- 71% to 80%

Avec un TYPE 1 , tournez le "ANT KNOB" produira:

u u c u u d d c d d

Si la déclaration etait :

ANT 6 5 (8 20 40 45 70 80) u d

Cela produira en Type 6 :

u u u u u d d d d d

6.2.6.1 Comprendre l attribut FORCE_MACROS

Reportez vous a la section 6.2.1.1 pour les explications sur les déclarations de TYPE 1 concernant cet attribut.

6.2.7 Sens des axes, valeurs analogiques et déclaration s digitales.

Dans cette section, nous vous montrerons comment et quand des valeurs digitales sont produites par des axes, et nous vous donnerons des exemples afin de clarifier chaque types de déclaration s digitales ,pour vous démontrer dans quelles directions les axes travaillent.

Position de l axe	Valeur analogique
JOYX - left	0
JOYX - right	max
JOYY - back	max
JOYY - forward	0
THR - back	max
THR - forward	0
RNG - CCW [note 1]	max
RNG - CW	0
ANT - CCW [note 1]	max
ANT - CW	0
MIX - left [note 2]	-
MIX - right	-
MIY- down	-
MIY - up	-
RDDR - left forward	0
RDDR - right forward	max
LBRK, RBRK - up	max
LBRK, RBRK - pressed	0

NOTES

1. *L utilisation du "Range knob" est souvent troublée par le sens de rotation que l on veut donne dans un déclaration digitale. Avec le* RNG *et* ANT *, la règle est la suivante: regardez le "Knob" de face pour déterminer le sens de rotation horaire et "ante-horaire".*
Horaire -> sens des aiguilles d une montre. ***CW***
Ante horaire > sens inverse des aiguilles d une montre. ***CCW***

2. *Le Microstick n est pas présenté a windows comme étant un contrôleur analogique. Mais cela ne veut pas dire qu il ne puisse pas être utilise comme tel.*

6.2.7.2 Déclaration s digitales d axe de TYPE 1

JOYX 1 6 r l	Quand l axe Joystick X est déplace de gauche a droite , nous obtenons le caractère "r" et de droite a gauche le caractère "l".
JOYY 1 6 f b	Quand l axe Joystick Y est déplace d arrière en avant , nous obtenons le caractère "f" et d avant en arrière le caractère "b".
THR 1 6 f b	Quand l axe Throttle est déplace d arrière en avant , nous obtenons le caractère "f" et d avant en arrière le caractère "b".
RNG 1 6 r l	Quand l axe RNG est tourne dans le sens CCW a CW, nous obtenons le caractère "r" et dans le sens CW a CCW le caractère "l".
ANT 1 6 r l	Quand l axe ANT est tourne dans le sens CCW a CW, nous obtenons le caractère "r" et dans le sens CW a CCW le caractère "l".
MIX 1 6 r l	Quand l axe Microstick X est déplacé de gauche a droite , nous obtenons le caractère "r" et de droite a gauche le caractère "l".
MIY 1 6 u d	Quand l axe Microstick Y est déplacé de bas en haut nous obtenons le caractère "u" et de haut en bas le caractère "d".
RDDR 1 6 l r	Quand la pédale da gauche est poussée, nous obtenons le caractère "l" et la pédale de droite le caractère "r".
LBRK 1 6 d u	Quand l axe Toe brake est pressé vers le bas nous obtenons le caractère "d" et lorsque relâché le caractère "u". (Même utilisation pour RBRK).

6.2.7.3 Déclaration s digitales d axe de TYPE 2

JOYX 2 5 a b c d e	Quand l axe Joystick X est déplacé de gauche a droite , nous obtenons les carate res "a b c d e" et de droite a gauche les caractères "e d c b a" .
JOYY 2 5 a b c d e	Quand l axe Joystick Y est déplacé d arrière en avant , nous obtenons les caractères "a b c d e" et d avant en arrière les caractères "e d c b a.
THR 2 5 a b c d e	Quand le Throttle est déplacé d arrière en avant, nous obtenons les caractères "a b c d e" et d avant en arrière les "e d c b a" .

RNG 2 5 a b c d e — Quand l axe RNG est tourne dans le sens CCW a CW, nous obtenons les caractères "a b c d e" et dans le sens CW a CCW les caractères "e d c b a" .

ANT 2 5 a b c d e — Quand l axe ANT est tourne dans le sens CCW a CW, nous obtenons les caractères "a b c d e" et dans le sens CW a CCW les caractères "e d c b a".

MIX 2 5 a b c d e — Quand l axe Microstick X est déplacé de gauche a droite , nous obtenons les caractères "a b c d e" et de droite a gauche les caractères "e d c b a".

MIY 2 5 1 2 3 4 5 — Quand l axe Microstick Y est deplace de bas en haut nous obtenons les caractères "1 2 3 4 5" et de haut en bas les caractères "5 4 3 2 1" .

RDDR 2 5 a b c d e — Quand la pédale de gauche est poussée et que nous la laissons revenir a sa position nous obtenons les caractères: "a b c d e". Quand nous poussons la pédale de gauche (et donc que celle de droite recule) nous obtenons les caractères "e d c b a" .

LBRK 2 5 a b c d e — Quand l axe Toe brake est pressé vers le bas nous obtenons les caractères "a b c d e", et lorsque relâche les caractères "e d c b a". (Même utilisation pour RBRK).

6.2.7.4 Déclaration s digitales d axe de TYPE 3

JOYX 3 l ^ r — Quand l axe Joystick X est déplacé a gauche, nous obtenons le continu "l" et a droite le caractère continu "r".

JOYY 3 b ^ f — Quand l axe Joystick X est deplace en arriere, nous obtenons le caractère continu "b" et en avant le caractère continu "f".

THR 3 b ^ f — Quand le Throttle est deplace en arriere, nous obtenons le caractères continu "b" et en en avant le caratcere continu "f" .

RNG 3 l ^ r — Quand l axe RNG est tourne dans le sens CW, nous obtenons le caractère continu "l" et dans le sens CCW le caractère continu "r".

ANT 3 l ^ r — Quand l axe ANT est tourne dans le sens CW, nous obtenons le caractère continu "l" et dans le sens CCW le caractère continu "r".

MIX 3 l ^ r — Quand l axe Microstick X est vers la gauche, nous obtenons le caratere continu "l" et vers la droite le caractère continu "r" .

MIY 3 d ^ u Quand l axe Microstick Y est deplace vers le bas, nous obtenons le caratere continu "d" et vers le haut le caractère continu "u" .

RDDR 3 l ^ r Quand la pedale de gauche est poussee nous obtenons le caratere : "l". Et Quand nous pedale de droite est poussee nous obtenons le caratere continu "r" .

LBRK 3 u ^ d Quand l axe Toe brake presse vers le bas nous obtenons le caractère continu "u", et lorsque relaché le caractère continu "d". (Meme utilisation pour RBRK).

6.2.7.5 Déclaration s digitales d axe de TYPE 4

JOYX 4 300 l ^ r Quand l axe Joystick X est deplace a gauche, nous obtenons le caractère repete "l" et a droite le caractère repete "r".

JOYY 4 300 b ^ f Quand l axe Joystick X est deplace en arriere, nous obtenons le caractère repete "b" et en avant le caractère repete "f".

THR 4 300 b ^ f Quand le Throttle est deplace en arriere, nous obtenons le caractères repete "b" et en en avant le caratcere repete "f" .

RNG 4 300 l ^ r Quand l axe RNG est tourne dans le sens CW, nous obtenons le caractère repete "l" et dans le sens CCW le caractère repete "r".

ANT 4 300 l ^ r Quand l axe ANT est tourne dans le sens CW, nous obtenons le caractère repete "l" et dans le sens CCW le caractère repete "r".

MIX 4 300 l ^ r Quand l axe Microstick X est vers la gauche, nous obtenons le caratere repete "l" et vers la droite le caractère repete "r" .

MIY 4 300 d ^ u Quand l axe Microstick Y est deplace vers le bas, nous obtenons le caratere repete "d" et vers le haut le caractère repete "u" .

RDDR 4 300 l ^ r Quand la pedale de gauche est poussee nous obtenons le caratere : "l". Et Quand nous pedale de droite est poussee nous obtenons le caratere repete "r" .

LBRK 4 300 u ^ d Quand l axe Toe brake presse vers le bas nous obtenons le caractère repete "u", et lorsque relaché le caractère repete "d". (Meme utilisation pour RBRK).

6.2.7.6 Déclaration s digitales d axe de TYPE 5

JOYX 5 5 (0 20 40 60 80 100) a b c d e Quand l axe Joystick X est deplace de gauche a droite , nous obtenons les

carateres "a b c d e" et de droite a gauche les caractères "e d c b a" .

JOYY 5 5 (0 20 40 60 80 100) a b c d e — Quand l axe Joystick Y est deplace d arriere en avant, nous obtenons les caractères "a b c d e" et d avant en arriere les caractères "e d c b a.

THR 5 5 (0 20 40 60 80 100) a b c d e — Quand le Throttle est deplace d arriere en avant, nous obtenons les caractères "a b c d e" et d avant en arriere les "e d c b a" .

RNG 5 5 (0 20 40 60 80 100) a b c d e — Quand l axe RNG est tourne dans le sens CCW a CW, nous obtenons les caractères "a b c d e" et dans le sens CW a CCW les caractères "e d c b a" .

ANT 5 5 (0 20 40 60 80 100) a b c d e — Quand l axe ANT est tourne dans le sens CCW a CW, nous obtenons les caractères "a b c d e" et dans le sens CW a CCW les caractères "e d c b a".

MIX 5 5 (0 20 40 60 80 100) a b c d e — Quand l axe Microstick X est deplace de gauche a droite , nous obtenons les carateres "a b c d e" et de droite a gauche les caractères "e d c b a" characters.

MIY 5 5 (0 20 40 60 80 100) a b c d e — Quand l axe Microstick Y est deplace de bas en haut nous obtenons les caractères " a b c d e " et de haut en bas les caractère " e d c b a " .

RDDR 5 5 (0 20 40 60 80 100) a b c d e — Quand la pedale de gauche est poussee et que nous la laissons revenir a sa position nous obtenons les carateres : "a b c d e". Quand nous poussons la pedale de gauche (et donc que celle de droite recule) nous obtenons les caractères "e d c b a" .

LBRK 5 5 (0 20 40 60 80 100) a b c d e — Quand l axe Toe brake presse vers le bas nous obtenons les caractères "a b c d e", et lorsque relaché les caractères "e d c b a". (Meme utilisation pour RBRK).

6.2.7.7 Type 6 Digital axes statements

JOYX 6 5 (0 20 40 60 80 100) r l	Quand l axe Joystick X est deplace de gauche a droite , nous obtenons le caractère "r" et de droite a gauche le caractère "l".
JOYY 6 5 (0 20 40 60 80 100) f b	Quand l axe Joystick Y est deplace d arriere en avant , nous obtenons le caractère "f" et d avant en arriere le caractère "b".
THR 6 5 (0 20 40 60 80 100) f b	Quand l axe Throttle est deplace d arriere en avant , nous obtenons le caractère "f" et d avant en arriere le caractère "b".
RNG 6 5 (0 20 40 60 80 100) r l	Quand l axe RNG est tourne dans le sens CCW a CW, nous obtenons le caractère "r" et dans le sens CW a CCW le caractère "l".
ANT 6 5 (0 20 40 60 80 100) r l	Quand l axe ANT est tourne dans le sens CCW a CW, nous obtenons le caractère "r" et dans le sens CW a CCW le caractère "l".
MIX 6 5 (0 20 40 60 80 100) r l	Quand l axe Microstick X est deplace de gauche a droite , nous obtenons le caractère "r" et de droite a gauche le caractère "l".
MIY 6 5 (0 20 40 60 80 100) u d	Quand l axe Microstick Y est deplace de bas en haut nous obtenons le caractère "u" et de haut en bas le caractère "d".
RDDR 6 5 (0 20 40 60 80 100) l r	Quand la pedale da gauche est poussee, nous obtenons le caractère "l" et la pedale de droite le caractère "r".
LBRK 6 5 (0 20 40 60 80 100) d u	Quand l axe Toe brake presse vers le bas nous obtenons le caractère "d" et lorsque relache le caractère "u". (Meme utilisation pour RBRK).

Cela couvre les differentes facons de programmer les nombreux axes de facons digitales. Nous allons aborder maintenant la partie analogique de ces axes et comment les affecter parmis les differents types de déclaration s.

La premiere chose que vous voulez certainement savoir est : commentchanger la courbe de reponse des axes a travers le sysyteme de programmation.

6.3 Courbes de reponse (CURVE)

Les 10 axes ont par defaut une courbe de reponse ***lineaire.*** Cela veut dire, par exemple que si vous poussez la "THROTTLE" vers l avant, les valeurs envoyees aux simulateurs seront directement en rapport avec la distance parcourue par la "THROTTLE".
Ex: chanques mouvements de 10 % de la "trottle" aura pour resultat une valeur de sortie en augmentation de 10%.
Il est possible de changer le comportement des 10 axes en changeant leurs courbes de reponses respectives. La courbe de reponse de chaques axes definie leurs sensibilites.
2 declaraitions peuvent etre utilisees pour changer la courbe de reponse des axes. Nous allons definir leurs syntaxes et ensuite voir comment les utiliser.

Déclaration de configuration

USE CURVE (Axis_Identifier, Sensitivity)

Syntaxe de la commande

CURVE [Slash modifiers] (Axis_Identifier, Sensitivity)

Ou:

Axis_Identifier est l un des suivants :

JOYX, JOYY	(tout deux appeles **JOYSTICK**)
THR	
RNG, ANT	(tout deux appeles **ROTARIES**)
MIX, MIY	(tout deux appeles **MICROSTICK**)
LBRK, RBRK	(tout deux appeles **TOEBRAKES**)
RDDR	

Sensitivity

Est une valeur comprise entre -32 et 32 (bien qu entre nous vous ne voudrez jamais utiliser une courbe resultant d une valeru superieura a 20 !o). Un nombre negatif (-10) par exemple represente une reduction de la sensibilite (ideal pour les atterrisages et les ravitaillements en vol et les vols en formations). Zero ,0 (neutre) reinitialise la courbe a son niveau de valeur lineaire par defaut (surpasse toutes les déclaration s de type : USE CURVE). Les nombres positfs produisent une augmentation de la sensibilite (ideal pour le combat aerien avec des vieux coucous de la seconde guerre).

Slash modifiers (attribut) *(optional)*

/U, /M, /D (Dogfight switch) et /I, /O (Bouton S3) sont permis.

Un peu perdu... moi aussi !o). Regardons quelques exemples et observons les differences entre les déclaration s USE CURVE et CURVE:
USE CURVE est une déclaration de configuration (toutes les déclaration s USE le sont) cela veut dire qu elle se definie dans sa propre ligne de commandes, pres du haut de la page du JOYSTICK FILES et elle ne peut etre programme dans une position du controleur, elle est normalement utilisee pour definr la courbe de reponse par **defaut** d un axe. Normalement chaque axe est lineaire, donc parametrer la courbe de reponse de l axe "JOYSTICK X" comme ceci :

```
USE CURVE (JOYX, 0)
```

Serait inutile car le compilateur le fait de lui meme de toutes les facons. Suposons que vous voulier une courbe de reponse lineaire sur l axe "JOYSTICK X" , si nous avions la ligne suivante dans le JOYSTICK FILES:

```
USE CURVE (JOYSTICK, 2)
```

Cela provoquerait la modification des 2 axes X et Y du JOYSTICkK de facon a etre plus sensible. Notez comment nous avons utilise le terme JOYSTICK pour definr les axes JOYX et JOYY , ce qui produira les 2 instructions suivantes pour le controleur.

```
USE CURVE (JOYX, 2)
USE CURVE (JOYY, 2)
```

Donc la déclaration "USE CURVE" peut etre utiliser pour modifier les courbes de reponses par defaut des axes. La déclaration "CURVE" suit la meme syntaxe, mais ce ne st pas une déclaration de configuration. Elle peut etre utiliser dans les déclaration de boutons, les déclaration s digitales d axes, ou dans sa propre déclaration, regardons les exemples ci-dessous :

```
CURVE  /U (JOYY, 2)   Rem  Plus sensible pour le combat
       /M (JOYY, 0)   Rem  Normale
       /D (JOYY, -2)  Rem  Moins sensible pour l atterrisage
```

De meme pour le Microstick:

```
CURVE  /I (MICROSTICK, 2)   Rem Plus sensible
       /O (MICROSTICK, 0) Rem  Normale
```

Et vous pouvez meme les melanger:

```
CURVE  /U /I (MICROSTICK, 2) (THR, 2)
         /O (MICROSTICK, 0)
```

```
/M /I (RDDR, -2)
   /O (RDDR, 0) (TOEBRAKES, 2)
/D (JOYY, -2)
```

La déclaration "CURVE" peut etre aussi directement utilise dans une position programmable:

```
BTN T7 /P CURVE(JOYX, 3)  Rem Plus sensible
       /R CURVE(JOYX, 0)  Rem  Normale
```

La déclaration "CURVE" peut etre aussi utilise avec une déclaration digital d axe, donc si nous voulions nous pourrions changer les coubes de reponses du JOYSTICK en fonction de la position de la "throttle":

```
THR 2 5 CURVE(JOYX, -3) CURVE (JOYX, -1) CURVE (JOYX, 0)
CURVE (JOYX, 2) CURVE (JOYX, 5)
```

Avec une valeur basse de la throttle l axe X du joystick est moin sensible , mais devient progressivement plus sensible avec l augmentation de la valeur retournee par le throttle.
C est un tres bon exemple de comment le Throttle peut etre gerer en analogique pendant qu il est programme de facon digitale, ce qui peut etre tres pratique.

NOTES

1. *Si un axe est defini en un autre axe (voir plus loin) alors sa courbe de reponse reste la meme.*

2. *Vous ne pouvez pas definir de "deadzones" avec la déclaration CURVE .Vous devez utiliser le panneau de configuration COUGAR CCP.Sivous avez besoin de "deadzones" pour un simulateur particulier, utilisez le CCP pour sauvegarder ces"deadzones" dans un profil et utilisez la déclaration USE PROFILE dans la JOSYTICK FILE , voir point 5.*

3. *Vous ne pouvez pas avoir plus d une déclaration CURVE dans votre JOYSTICK FILE .Donc (dans cet exemple) la seconde déclaration provoquera une erreu de compilation..*

```
CURVE  /I (MICROSTICK, 2)   Rem Plus sensible
       /O (MICROSTICK, 0) Rem  Normale

CURVE  /I (ROTARIES, 2)   Rem Plus sensible
       /O (ROTARIES, 0) Rem  Normale
```

A ne pas confondre avec l utilisation de CURVE dans la programmation des axes et des bouttons, qui vous permet d en utiliser plus qu un seul.

4. *Si vous n utilisez pas CURVE dans une déclaration de bouton ou d axe, alors vous ne pourrez l utiliser seul sans attribut (ex: /M SLASH MODIFIER. A la place utilisez une déclaration de configuration : exemple :*

CURVE (JOYSTICK, 10)	*generera une erreur de compilation alors que :*
USE CURVE (JOYSTICK, 10)	*est correcte.*

5. *Vous pouvez autrement utiliser un profil sauvegarde (voir section USE PROFILE plus haut) si vous appliquez plusieurs courbes de reponses a des axes pour l integralite du fichier .Cela a pour avantages de pouvoir incorporeregalement des DEADZONES*

6.4 TRIM d axes (TRIM)

"Trimer" un axes veut dire : pouvoir enlever les mains des commandes et faire croire aux simulateurs que vous avez toujours les mains dessus et que vous exercez une pression. Un petit exemple pour mieux comprendre :
Imaginons que vous voliez en croisiere a 15000 ft et pour certaines raisons (vent, poids) l avion veut monter tout seul, meme quand les commandes sont centrees, vous devez donc corriger en permanence en poussant sur le joystick. La fonction TRIM vous permet de relacher les commandes et de faire croire aux simulateurs que vous poussez toujours sur le joystick. Ce n est pas limite au joystick mais peut s appliquer au 10 axes analogiques.

Syntaxe de la commande
TRIM (Axis_Identifier, Trim_amount)
et:
HOLDTRIM (Axis_Identifier)

ou :

Axis_Identifier est l un des suivants :

JOYX, JOYY	(tout deux appelles **JOYSTICK**)
THR	
RNG, ANT	(tout deux appelles ROTARIES)
MIX, MIY	(tout deux appeles MICROSTICK)
LBRK, RBRK	(tout deux appeles TOEBRAKES)
RDDR	

Trim_Amount

Est une valeur qui varie de -128 a 127 ou TO_CURRENT.

Une valeur positive augmentera la valeur du trim alors qu une negative la diminuera. Une valeur 0 reinitialisera la courbe de reponse de l axe, il en resultera qu aucun trim ne sera plus applique.Une valeur positive a La commande TO_CURRENTprend les valeurs en cours de l axe et les applique comme valeurs de trim lorsque l axe est centre

Bon, prenons un exemple qui utilise les axes RANGE et ANT pur ajuster le trim sur les axes X et Y du Joystick, nous utiliserons une delcaration de TYPE 1:

```
RNG 1 12 TRIM (JOYX, 20+) TRIM (JOYX, 20-)
ANT 1 12 TRIM (JOYY, 20-) TRIM (JOYY, 20+)
```

Tourner le ANT dans le sens horaire (CW) par exemple va soustraire 20 de la valeur de l axe Y , ce qui equivaut a pousser le Joystick en avant. Ce qui est res utile pour stabiliser un avion qui monte quand les commandes sont centrees. Nous pouvons definir le bouton S2 sur le Joystick pour annuler les effets du trim, comme cela :

```
BTN S2 TRIM (JOYX, 0) TRIM (JOYY, 0)  Rem Supprime le trim des 2 axes
```

Ou egalement par cette ligne (memes effets):

```
BTN S2 TRIM (JOYSTICK, 0)
```

Nous pouvons aussi specifier des valeurs a affecter

```
BTN S4 TRIM (JOYX, 5) TRIM (JOYY, -10)
```

De plus je peux maintenir le Joystick dans une position donnée et regler le trim pour q une fois le Joystick relaché, il garde les valeurs du trim pour la position definie :

```
BTN S2 /I TRIM (JOYSTICK, TO_CURRENT) Rem Trim au valeurs en cours
       /O TRIM (JOYSTICK, 0) Rem annule les trim
```

Je pense que la vous rencontrez un petit probleme.. non ???
Remarquez que j ai precisé "quand le Joystick est relaché" .Laissez moi vous expliquer:
Si vous etes en vol de croisiere et que vous maintenez le Joystick pousse en avant (loin de sa position centrale) afin de garder l avion au meme niveau (sans monter ni descendre) car votre avion a tendance a cabrer tout seul , et que maintenant vous "trimiez" votre Joystick aux valeurs de sa position actuelle.Vous vous attendez surement a pouvoir ramener le Joystick en position centrale et voir votre avionrester enn pallier. Mais ce n est pas ce qui va se passer avec la déclaration TRIM (vue au dessus) a moins que vous ne relachiez le Joystick au moment meme ou vous reglez le trim . Pourquoi ??? Car la valeur que vous voulez attribuer au réglage du trim est calculée en supposant que le Joystick soit dans sa position centrée .Mais dans ce cas ci il n y est pas car vous poussez toujours le Joystick en avant pendant que vous réglez le trim. ....

Des que vous allez régler le trim. , cela simulera la poussée du Joystick vers l avant, les deux réglages vont se cumuler et l avion va plonger ! Des que vous replacez le Joystick en position il semblera aux simulateurs que vous poussez le manche de façon a maintenir l avion a niveau (but initialement recherche). Pour une bonne compréhension vous devrez probablement relire ce paragraphe plusieurs fois (je l ai fait moi même !o)

Alors comment pouvons nous éviter cela : Nous avons 2 manières d y arriver la facile et la difficile
Voici la facile :

```
BTN S2 HOLDTRIM (JOYSTICK)
```

La raison pour laquelle nous allons également voir la manière difficile est que cela permet de comprendre facilement l utilisation de cette déclaration !o)
La méthode employée est : Maintenez le Joystick dans la position ou voter avion vole en pallier. Maintenant appuyez sur le bouton S2 puis ramenez votre Joystick en position centrale et maintenant et seulement maintenant relâchez le bouton S2.
Aussi longtemps que vous appuyez sur le bouton S2 (et pendant que vous bougez votre Joystick) votre avion continuera à voler au même niveau. Une fois le Joystick replace en position centrale vous pouvez relâcher S2 et enlever vos mains du Joystick.

Maintenant voyons la manière d implémentation difficile :o)
En fait cela n est pas aussi difficile que cela et cela aide beaucoup à comprendre ce que "fait" la déclaration située plus haut. Nous devons utiliser la déclaration TRIM TO_CURRENT en combinaison avec les déclaration s LOCK et UNLOCK (*voir notes plus loin*), comme cela :

```
BTN S2 /P LOCK (JOYSTICK, LASTVALUE) TRIM(JOYSTICK, TO_CURRENT)
       /R UNLOCK (JOYSTICK)
```

Maintenant pendant que je pousse le Joystick vers l avant pour maintenir un niveau de vol stable, si j appuie et que je garde appuyé le bouton S2, l avion va maintenir son niveau de vol tandis que je ramène le Joystick en position centrale et une fois en position je relâche le bouton S2.
Alors comment cela fonctionne t il ?
La premier chose qui se passe quand je presse sur le bouton S2 est que le Joystick est verrouillé a ses valeurs actuelles, et dans la foullée le TRIM est calcule a partir de ces "valeurs bloquées"
Quand le Joystick est repositionne dans sa partie centrale nous déverrouillons le Joystick en relâchant S2 et l avion vole toujours au même niveau car nous avons déjà "trimé" les axes. C est effectivement de cette manière que BTN S2 HOLDTRIM (JOYSTICK) est traduit par le compilateur.

NOTES

1. Un changement de valeurs de TRIM résulte d une addition ou d une soustraction d une valeur donnée a la courbe de réponse d un axe. Cela déplace l ensemble de la courbe dans une direction ou vers une autre. Cela na pas d importance que vous utilisiez une courbe linéaire ou une courbe de réponse modifiée.

2. Une courbe linéaire "trimée" ne vous permettra pas d utiliser la course complète de l axe.

3. Inverser un axe ne modifie pas la direction de la fonction TRIM, elle reste la même. Les déclarations digitales ne s inversent pas avec leurs équivalents analogiques.

4. Faites attention ou vous placez les signes + et – dans une déclarations d axes. Du coté gauche ils précisent les valeurs du TRIM, du coté droit du nombre l opération (addition ou soustraction) de la valeur définie du TRIM. Consultez la section "comprendre la souris et le Microstick" pour comprendre un peu mieux les différences entre les signes + et - leurs effets suivant leurs positions a droite ou a gauche de la valeur.
Vous pouvez aborder cela tranquillement et précisément avec le logiciel "*COMPOSER*".

5. Ces déclarations HOLDTRIM sont toutes valides

```
BTN T6 a b HOLDTRIM (RNG) c d
BTN S4 /P a HOLDTRIM (RNG)
       /R b
BTN S1 a { HOLDTRIM (JOYY) b HOLDTRIM (ANT) }
```

Remarquez que les déclaration s multiples HOLDTRIM (comme ci-dessus) doivent être regroupées entre des " *curly brackets"* ***{ }***

6. Vous ne pouvez pas avoir de macros appelées TRIM, mais vous pouvez avoir Ttrim ouTrim _Hold par exemple.

7. Vous pouvez utiliser l attribut "AUTOREPEAT" (/A) en association avec la déclaration TRIM pour contrôler n importe lesquels des axes avec n importe lesquels des boutons ou "HAT" .
Exemples de déclaration s "trimant" les axes du Joystick.

```
BTN H1U /A TRIM (JOYY, 5-) DLY(120)
BTN H1D /A TRIM (JOYY, 5+) DLY(120)
BTN H1L /A TRIM (JOYX, 5-) DLY(120)
BTN H1R /A TRIM (JOYX, 5+) DLY(120)
```

Il est également possible d adresser des axes qui ne sont pas physiquement présents (palonniers freins....) en utilisant leurs déclarations respectives (consultez le manuel de référence, le chapitre sur l activation et désactivation des axes windows a l aide du panneau de contrôle Cougar. Système utilisant les "Checkbox").
Par *exemple :*

BTN H4L /A TRIM (RDDR, 5-)
BTN H4R /A TRIM (RDDR, 5+)

6.5 Désactivation d'Axes

Tout les axes analogiques sont signales aux simulateurs comme présent par défaut, a exception des axes du Microstick .Il existe quelques circonstances dans lesquelles il est préférable pour certains d être désactivé avant le lancement du simulateur, comme celui ou vous voulez seulement les utiliser pour produire des caractères clavier au moyen d une déclaration digitale.
Les axes peuvent être désactivés avec une déclaration de configuration située dans le JOYSTICK FILE .

Déclaration de configuration
DISABLE Axis_Identifier

Ou :

Axis_Identifier est :

THR	
RNG, ANT	(tout deux appelles **ROTARIES** –voir plus bas)
LBRK, RBRK	(tout deux appelles **TOEBRAKES** – voir plus bas)
RDDR	

En outre il peut être utile de pouvoir définir un groupe de ces axes qui pourront par la suite être désactivés avec une seule ligne de déclaration :
Exemple :

DISABLE ROTARIES	est converti par le compilateur en	DISABLE RNG DISABLE ANT
DISABLE TOEBRAKES		DISABLE LBRK DISABLE RBRK

Comme toutes les déclarations de configuration cela apparaît sur leur propre linge de commande dans le "JOYSTICK FILE", l attribution "REM" permet d inclure un descriptif.
Donc :

DISABLE THR Rem désactive le throttle
DISABLE ANT Rem désactive l antenne knob sur la TQS

Sont correctes alors que :

DISABLE (THR, ANT, RNG) *ne l est pas.*

Cela peut paraître trop propre mais il y a des avantages pour les programmeurs et les utilisateurs à les garder séparés sur leur propre ligne. Il est plus simple d annoter (rem) une seule commande par ligne. Remarquez que la nouvelle déclaration "DISABLE" remplace l ancienne déclaration "USE NO" de l ancienne syntaxe TM (USE NO MOUSE ...).
Comme tous les axes sont présent par défaut, vous n avez pas besoin d utiliser les déclarations comme USE RCS, USE TQS, comme sur les anciens HOTAS TM.

6.5.1 Activation et désactivation d axes en vol avec LOCK, UNLOCK

Nous savons comment désactiver un axe en utilisant une déclaration de configuration. Mais une fois l axe désactivé, il ne peut plus être "vu" par le simulateur comme étant un axe analogique. Donc cela ne sert a rien a moins d être programmé digitale ment.
Maintenant il y a parfois des circonstances vous voudriez par exemple pouvoir bouger un axe analogique en utilisant sa programmation digitale mais sans changer ses valeurs analogique.
Malheureusement il n est pas possible d enlever ou d ajouter des axes une fois dans le jeu, car cela produirait souvent des effets comme le gel ou le plantage de votre jeu.
Donc si nous ne pouvons enlever un axe en vol et le remettre par la suite , la seule façon de contourner ce problème et de maintenir l axe a une certaine valeur définie et de bouger cet axe en l exploitant cette fois de façon digitale.
Nous pouvons faire cela avec la déclaration "LOCK" et d activer de nouveau l axe avec la déclaration "UNLOCK".
Voici la syntaxe :

Syntaxe de la commande
LOCK (Axis_Identifier, Lock_Value%)
UNLOCK (Axis_Identifier)

Ou :

Axis_Identifier est :

JOYX, JOYY	(tout deux appelles JOYSTICK)
THR	
RNG, ANT	(tout deux appelles ROTARIES)
MIX, MIY	(tout deux appelles MICROSTICK)
LBRK, RBRK	(tout deux appelles TOEBRAKES)
RDDR	

Lock_Value est:

Soit de 0 a 100%, ou simplement LASTVALUE

Par exemple :

```
BTN S2 /I LOCK (THR, 100%)
       /O UNLOCK (THR)
```

Donc la déclaration "LOCK" est utilisée pour forcer un axe à générer une valeur seule dans une "zone définie" ou sur la totalité de sa course, ou de générer une valeur finale avec la syntaxe "LASTVALUE". La nature analogique de l axe peut être restaure par l utilisation de la déclaration "UNLOCK".

Un peu perdu ? La meilleure façon de comprendre est d'examiner quelques exemples :

1. Admettons que nous voulions utiliser "RANGE KNOB" de façons analogiques <u>sauf</u> lorsque le bouton S3 est pressé afin qu il exécute une déclaration digitale, tout en maintenant la dernière valeur du "RANGE KNOB".

```
RNG /I LOCK (RNG, LASTVALUE) 2 5 a b c d e
    /O UNLOCK (RNG)
```

Donc quand le bouton S3 n est pas pressé le "RANGE KNOB " est reconnu comme un axe analogique standard et traite comme tel par le jeu.
Quand le bouton S3 est presse la valeur du "RANGE KNOB" ne change plus, il mémorise la dernière valeur analogique de sortie et permet la génération de caractères "abcde" de la déclaration de TYPE 2 grâce a l attribut /I .
J espère que cela est suffisamment clair, mais observons juste comment cette commande peut être puissante :

2.

```
ANT  /U /I LOCK (ANT, LASTVALUE) 2 10 1 2 3 4 5 6 7 8 9 0
        /O UNLOCK (ANT) Rem axe de jeu assigné
     /M UNLOCK (ANT) Rem axe de jeu assigné
     /D LOCK (ANT, 0%) 3 Lower_flaps ^ Raise_Flaps
```

Lorsque le switch"DOGFIGHT" est placé en position "DOWN" (bas) (/D) le "ANT KNOB" est utilisé de manière digitale pour la gestion des "volets" au moyen d'une déclaration de type 3, et le jeu recevra une valeur zéro pour la commande qu'il a assigné à l "ANT KNOB".
Quand le switch "DOGFIGHT" est placé en position MDDLE milieu (/M) l "ANT KNOB" réagira directement comme programmé dans le jeu. Enfin lorsque le "SWITCH DOGFIGHT" est en position "HAUTE (UP) (/U)", et si le bouton S3 n'est pas pressé, le ANT KNOB sera géré comme contrôle analogique, mais si le bouton S3 est pressé la dernière valeur analogique sera mémorisé et les nombres 1 à 10 seront générés lorsque le "ANT KNOB" sera tourné.

Utiliser Lock et Unlock en combinaison avec des déclarations numériques permet une grande souplesse et capacités de programmation sur tous les axes. Mais attention le risque d'erreur de programmation augmente de même.

NOTES

1. Vous ne pouvez pas désactiver les axes du Joystick ou du Microstick avec la déclaration de configuration DISABLE. Les axes de joystick doivent être actifs, c'est une obligation de DirectX.

2. Quant vous désactiver un axe dans un fichier de configuration, le contrôleur doit effectuer une série de test par lesquels windows est informé de l'absence d'un axe. Cela peut demander un peu plus de temps, et charger un tel fichier de configuration va augmenter le nombre de "rapport" à windows et cela prendra un peu plus de temps.

.

3. Les déclarations LOCK *et* UNLOCK doivent être précédés des attributs /U, /M, /D, /I ou /O, quand ils sont utilisés dans des déclarations d'axes. Sinon cela génèrera une erreur de compilation de code.

ANT LOCK (RNG, LASTVALUE)

Mais : BTN S2 LOCK (RNG, LASTVALUE) *est correcte !*

6.6 AFFECTATION DES AXES (SWAP)

L'affectation des axes vous permet d'intervertir des axes entre eux, avant le décollage en vol par exemple. Cela est possible grâce à la déclaration SWAP déclaration de configuration.

Déclaration de configuration

USE SWAP (Axis_Identifier, Axis_Identifier)
Syntaxe de la commande SWAP (Axis_Identifier, Axis_Identifier)

Ou :

Axis_Identifier est l'un des suivants :

JOYX, JOYY	(tout deux appelles **JOYSTICK**)
THR	
RNG, ANT	(tout deux appelles **ROTARIES**)
MIX, MIY	(tout deux appelles **MICROSTICK**)
LBRK, RBRK	(tout deux appelles **TOEBRAKES**)
RDDR	

Par exemple :

```
USE SWAP (ANT, RNG)
```

Il en résulte que les axes "ANTENNA" et "RANGE" situés sur la manette des gaz seront intervertis :
D'autres exemples

```
BTN S1 SWAP(JOYY, THR)          REM interverti l axe Y du Joystick et l axes
                                de la Throttle
BTN S2 SWAP(JOYSTICK, MICROSTICK)    REM interverti les axes Joystick
                                X and Y avec les axes Microstick X
                                and Y
```

Le dernier exemple est converti comme cela le compilateur :

```
BTN S2 SWAP(JOYX, MIX) SWAP(JOYY, MIY)
```

Et donc c'est important de remarquer que si vous intervertissez plusieurs axes à la fois vous ne pourrez le faire qu'avec le même nombre d'axes :

BTN S2 SWAP(JOYSTICK, MICROSTICK) est correcte mais

BTN S2 SWAP(JOYSTICK, THR)

Provoquera une erreur du compilateur, car joystick est défini par 2 axes JOYX et JOYY) alors que THR n'en a qu'un seul.

NOTES

Intervertir des axes intervertir également leurs courbes de réponse.
However any digital statements on those axes do not get swapped over.

6.7 Inverser la direction d'un axe (REVERSE, FORWARD)

Il est possible d'inverser la direction par défaut d'un axe en utilisant la déclaration REVERSE

Déclaration de configuration
USE REVERSE (Axis_Identifier)

Syntaxe de commande
REVERSE (Axis_Identifier)
FORWARD (Axis_Identifier)

Où
AXIS-Identifier est un des suivants

JOYX, JOYY	(tout deux appelles **JOYSTICK**)
THR	
RNG, ANT	(tout deux appelles **ROTARIES**)
MIX, MIY	(tout deux appelles **MICROSTICK**)
LBRK, RBRK	(tout deux appelles **TOEBRAKES**)
RDDR	

Cela peut être très pratique pour un simulateur d'hélicoptère par exemple, si vous voulez utiliser la manette des gaz dans la direction opposée à celle utilisée pour les simulateurs de Jet, ou si vous voulez que vos palonniers fonctionnent dans le sens inverse des avions réels.

Si je veux changer la direction par défaut de mon palonnier, sans avoir besoin d'appuyer sur un bouton je peux l'obtenir en utilisent **USE** devant la déclaration **Reverse**, dans une déclaration de configuration comme suivant :

```
USE REVERSE (RDDR)
```

Cela apparaîtra, presque au début du"joystick files" sur sa propre ligne :
Un autre exemple :

```
BTN H1U   /I REVERSE (JOYY)
          /O FORWARD (JOYY)
```

Presser ment 1 UP et S3 provoquera l'inversion de l'axe joystick y, presser ment 1 UP sans le bouton S3 rétablira l'axe joystick y dans son sens normal. Notez que comme pour les autres propriétés de programmation des axes, les effets de REVERSE et de FORWARD restent avec les axes en cas de réaffectations ou d'autre changements. Mais remarquez, aussi que les déclarations digitales ne peuvent pas être inversées.

6.8 La déclaration : USE AXES_CONFIG

Nous avons parcouru en détails la façon de désactiver, intervertir et inverser des axes au moyen de déclaration de bases.
Il est aussi possible de faire tout cela à travers une déclaration de configuration utilisant "USE AXES_CONFIG" dans une programmation des joysticks. Le compilateur les convertira en déclaration de configuration, de même type que celle expliquées au dessus.

C'est une fonction complexe et rarement utilisé, mais qui peut être utile dans certaines circonstances.

Regardons la syntaxe en premier.

Déclaration de configuration :

USE AXES_CONFIG **(***DX-axis1*, HOTASaxis1), (*DX-axis2*, HOTASaxis2) etc.

ie. DX-axis1 is assigned to HOTASaxis1

Alors quelle est la différence entre DX-AXIS et HOTAS AXIS ?
Le denier et le plus simple à expliquer donc nous la ferons en premier.
Un HOTAS AXIS est un des 10 axes physiques du Cougar (JOYX, JOYY, THR, RDDR, ANT, RNG, MIX, MIY, LBRK, RBRK). Un DX-AXIS est un peu plus difficile à expliquer.

C'est un axes que le COUGAR rapporte à DIRECTX comme étant présent, auquel un jeu peut assigner une fonction. Maintenant bien que nous ayons 10 axes disponibles, directx8 n'en supporte que 8 via le port USB (DX 7 seulement 6). Ce que nous avons fait pour vous rendre la vie plus facile et de dire à windows "utilise l'axe JOYSTICKX en tant qu'axe DX-AXIS 1, JOYSTICK y en tant que axe DX-AXIS2 etc... Il ne sait pas si vous avez une manette de gaz, un RNG etc... Car il utilise ces axes même si vous avez un volant ou un autre type de contrôleur de jeu. Les "DX-AXES" sont définis avec les identifiants ci dessous puis assignés au COUGAR en tant que :

DirectX axis	HOTAS assigned axis	Syntaxe
X axis	Joystick's X axis	JOYX
Y axis	Joystick's Y axis	JOYY
Z axis	Throttle	THR
Rotation X	Throttle's Antenna knob	ANT
Rotation Y	Rudder's Left Toe brake	LBRK
Rotation Z	Rudder	RDDR
Slider 0	Throttle's Range knob	RNG
Slider 1	Rudder's Right Toe brake	RBRK

Notez, le freinage différentiel sera réimplanté lors de la production du palonnier COUGAR.

Notez, le freinage différentiel sera réimplanté lors de la production du palonnier COUGAR.

Revenons à l'utilisation de cette déclaration, il y a 4 règles de base à se rappeler :
1. Tous les axes (Directx ou COUGAR) transmis à windows seront désactivé de façon analogique
2. les axes entres crochets (...) sont assignés entre eux

3. chaque axes du COUGAR précédés d'un signe "-" seront inversés
4. vous devez avoir les axes 1 et 2 présents, c'est une obligation de windows.

Exemple :

USE AXES_CONFIG (1, RNG), (2, ANT), (3, -THR)

Dans cet exemple :

1. Le "RANGE KNOB" est assigné à l'axe 1 directx 1 et donc il sera reconnu comme l'axe Joystick x
2. Le "ANT KNOB" est assigné à l'axe DIRECTx 2 et donc il sera reconnu comme l'axe Joystick y
3. La manette des gaz "THR" est assigné à l'axe Directx 3, ce qui est normal, mais le "-" devant signifié que cet axe est inversé, utile pour les simulations d'hélicoptère
4. Aucuns autres axes n'est transmis à windows et ne seront donc pas disponibles durant le jeu, mais ils peuvent être heureusement programmés de façons digitales.

Vous remarquez q'avec une seule déclaration il est possible d'intervertir, de désactiver et d'inverser des axes et tout cela en une seule ligne de commande.

NOTES

1. Vous ne pouvez pas utiliser les déclarations DISABLE, USE REVERSE *ou* USE SWAP en conjonction avec USE AXES_CONFIG. Le faire provoquera une erreur de compilation. Mais sur une déclaration de "boutons" Vous pouvez toujours utiliser SWAP et REVERSE, si des axes sont présents.

2. Il est beaucoup plus facile d'utiliser "USE PROFILE en conjonction avec le panneau de contrôle du COUGAR.

3. Direct x établi sa propre assignation des axes qui lui sont rapportés présent, donc quand vous utilisez le panneau de control COUGAR vous pouvez parfois trouvez une assignation qui ne vous convient pas. Le seul moyen d'obtenir des résultats à votre convenance .... est de faire vos tests.

7. Programmation de la souris

7.1 Comprendre la souris et le MICROSSICK

Dans la partie finale de ce chapitre, nous verrons comment la souris marche et je vous montrerais des choses sympathiques que l'on peut faire avec.
Mais je ne vais pas, comme nous l'avons fait avant, vous montrez directement les déclarations et syntaxe associées. Car nous devons en savoir un peu plus sur le fonctionnement de la souris.

Je suis sure que vous avez déjà essayé le microstick sur le TQS et que vous l'avez assimilé à un contrôleur de souris, car cela fait bouger la souris.

Mais il y a un point à bien comprendre.
Le Microstick n est pas une souris.

Le microstick est comme un mini joystick, il contrôle le plus souvent la souris car le compilateur lui dit de le faire. Le compilateur assigne à ses deux axes (MIX et MIY) le contrôle de la souris

Alors qu'est ce que la souris ? Et pourquoi ne pouvons nous pas juste assigner les axes X et Y de la souris aux axes X et Y du microstick ?

La raison est la suivante, la souris n'est pas définie par des axes figés.
Bon, prenons le joystick quand nous le bougeons en cercle, cela génère des valeurs fixes par des coordonnées X et Y. Si le joystick contrôlait un curseur comme il le fait dans l'analyseur de joystick Foxy, son mouvement correspondrait au mouvement du joystick. Si le joystick s'arrêtait de bouger, le curseur s'arrêterait également. Mais le micro..... Est programmé avec un comportement de souris et non de joystick. Si vous le bouger comme un joystick et que vous le maintenez dans une position non centrale, le curseur continuera de bouger. Il ne s'arrêtera pas tant que le MICROSTICK ne soit pas revenu en position centrale. Cela vient du fait que le microstick ne dit pas à l'ordinateur : "ho!! Bouges le curseur à ces coordonnées X et Y et arrêtes toi" mais il lui dit plutôt "continus à faire bouger le curseur à la vitesse 3 sur l'axe X et à la vitesse 2 sur l'axe Y jusqu'à ce que je le dise autre chose". De plus quand le microstick rentre en position centrale, le curseur ne revient pas au centre de l'écran mais reste sur sa position car le microstick ce change ses instructions en : "OK arrêtes de bouger le curseur sur les axes X et Y".

Donc on fait bouger le curseur de la souris en assignant une valeur non nulle (différente de 0) a l axe MOUSSEX et/ou MOUSSEY (MSY et MSX). Et si nous programmons le MICROSTCK pour qu'il puisse changer les valeurs de MSX et MSY, alors le MICROSTICK contrôlera la souris

Et j'espère que vous commencez à comprendre la raison de ce style d'implantation. Nous pouvons utiliser n'importe quoi pour ajuster MSX et MSX. Le microstick, le joystick, le Hat, un bouton, logical flag....

…. Et ils peuvent le faire en même temps. Nous pouvons programmer le microstick pour des mouvements rapides de la souris et le Hat pour des mouvements plus fins.

7.2 USE MTYPE, la façon la plus simple d'assigner la souris au Microstick.

Plus tard dans ce chapitre, je vous expliquerai comment configurer la souris par le MICROSTICK et obtenir la sensibilité exacte que vous en attendez. Mais pour faire cela vous devez comprendre l'usage des déclarations de type digitale utilisant MSY et MSX.
Dieu merci il y a quelques déclarations qui peuvent très facilement assigner la souris au Microstick. Ce sont les déclarations USE MTYPE et USE MICROSTICK AS MOUSE.
Commençons par la commande USE MTYPE, c'est très simple à comprendre et à utiliser..

Déclaration de configuration :

USE MTYPE Type - REVERSE_*type*

Ou :

Type: est de A1 à A5 et décris quels boutons sur la manette des gaz vont être utilises en tant que boutons gauche et droite comme suivant .:

Type	bouton gauche de la souris	bouton droit de la souris
A1	T1	T6
A2	T6	T1
A3	T1	none
A4	T6	none
A5	none	none

T1 est le bouton intégré au Microstick. Pressez le microstick pour l'activer et T6 est le bouton du " range Knob"

REVERSE_*type* est REVERSE_UD et/ouREVERSE_LR

La commande REVERSE_UD inverse les axes haut et bas de la souris (axes y) et REVERSE_LR inverse les axes gauche et droite de la souris (axe x).

Exemples: USE MTYPE A3

Assigne la souris au microstick et le bouton gauche au bouton T1

USE MTYPE A5 - REVERSE_UD

Assigne la souris au microstick inverse la direction de l'axe Y et n'assigne aucun boutons

---

NOTES

1. La sensibilité de la souris est réglée par le compilateur qui la règle par défaut a une valeur correcte pour l utilisation sur un écran de résolution 1024 x 768 *.Vous pouvez modifier cette sensibilité par la commande* USE MTYPE *, donc si vous utilisez une plus haute résolution ou si vous voulez plus de sensibilité, alors vous devez utilisez la commande* USE MICROSTICK AS MOUSE *que nous verrons dans le prochain chapitre..*

2. *Le microstick comme nous l'avons dit avant est un contrôleur analogique. Ce n'est pas qu'un contrôleur 4 boutons, comme sur les HOTAS TM originaux, et de plus, les boutons T11 à T14 n'existe plus pour la programmation. Vous pouvez émuler T11 à T14 avec les instructions appropriées si vous voulez. Voir l'aide du fichier d'aide Foxy intitulé : "Instructions de conversion TQS T11-T14 pour l'utilisation avec le Stick Cougar". Souvenez vous que le Microstick est juste cela : un contrôleur avec 2 axes, pas une série de boutons. Il est bien plus puissant de cette façon.*

3. *Si vous attribuez une courbe quelconque au Microstick, cela n'affectera pas la souris, Comme s'il était attribué comme une instruction digitale, et les instructions digitales ne sont pas affectées par les courbes analogiques.*

4. *Si vous utilisez* USE MTYPE *dans votre fichier joystick, qui configure les boutons de souris sur T1 ou T6, alors vous ne pouvez pas programmer ces positions. Donc disons que nous avons :*

USE MTYPE A3

Que le compilateur, invisible pour vous, configure aussi l'instruction :

BTN T1 /H MOUSE_LB

Maintenant, vous avez n'importe où dans votre fichier :

BTN T1 *somemacro* ou BTN T1 /H MOUSE_RB

Alors le compilateur produira une erreur. Si vous voulez utiliser l'instruction USE MTYPE *pour configurer la souris sur le microstick, mais que vous voulez*

programmer T1 et/ou T6 séparément, alors utiliser USE MTYPE A5 ou une instruction appropriée qui n'assigne aucun bouton souris sur le bouton de la manette que vous voulez programmer.

7.3 UTILISER LE MICROSTICK COMME SOURIS

La seconde façon d'attribuer la souris sur le microstick, c'est avec l'instruction USE MICROSTICK AS MOUSE. Cela a l'avantage de plus que l'instruction USE MTYPE de pouvoir changer le comportement par défaut de la souris, ex. la vitesse de déplacement. Cela attribue aussi le bouton gauche à T1 sur le microstick par défaut, bien que vous puissez l'annuler avec le modificateur NO_BUTTON. Pour la plupart des simus bien sur, nous voulons attribuer le bouton gauche de la souris sur T1. L'instruction USE MICROSTICK AS MOUSE informe effectivement le compilateur de configurer soit des instructions digitales de Type 6 pour la souris sur le microstick, ou si une valeur de départ est fournie, une instruction de Type 5.
Regardons alors la syntaxe :

Déclaration de configuration :

Pour les instructions de Type 6 : *(Pas de valeur de départ fournie)*

Modificateur : USE MICROSTICK AS MOUSE (Scale value, Increment value)

Pour les instructions de Type 5 : *(Valeur de départ fournie)*

USE MICROSTICK AS MOUSE (Scale value, Increment value, Starting value)

*(**Note**: L'instruction USE MICROSTICK AS MOUSE ( ) attribue aussi le bouton gauche de la souris au bouton T1 sur le microstick, à moins qu'un modificateur NO_BUTTON est présent.)*

où :

Scale value: Nombre entre 2 et 12. Cela détermine en combien de plages, l'axe du microstick est divisé.

Increment value: Valeur d'augmentation de chaque plage, de 1 à 63. *(Notez que le Scale value multiplié par l' Incrément value doit être inférieur à 128. Ne pensez même pas à demander pourquoi ?!)*

Starting value: Valeur de départ pour une instruction de Type 5, pour laquelle s'applique la valeur d'augmentation. C'est la vitesse initiale de déplacement de la souris quand le microstick est déplacé de sa position centrale.

Modificateur :

REVERSE_UD:	Inverse l'axe Y du Microstick.
REVERSE_LR:	Inverse l'axe X du Microstick.
NO_BUTTON:	Informe le compilateur de ne pas configurer le bouton T1 comme bouton souris gauche. Cela vous autorise à programmer le bouton T1 avec l'instruction BTN T1.

Traitons directement avec quelques exemples :

USE MICROSTICK AS MOUSE (12, 2)

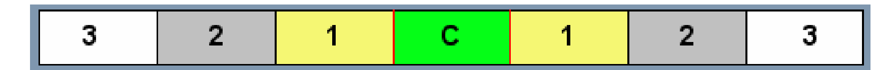
Cette instruction attribuera la souris sur le microstick, et configurera T1 comme bouton souris gauche. Regardons ces deux instructions :

USE MICROSTICK AS MOUSE (6, 4)
USE MICROSTICK AS MOUSE (7, 3, 2)

Celles là aussi attribueront la souris au microstick, et configureront T1 comme bouton souris gauche. Donc voici 3 instructions, qui sont totalement différentes, mais comme j'ai dit, ont exactement la même fonction. Eh bien, je disais vrai en disant qu'ils faisaient le même *en terme d'attribution* de la souris au microstick, mais ils sont différents quant à la réponse de la souris que vous verrez sur le microstick. Ce qui signifie bien sur que je dois essayer d'expliquer ce que font les valeurs entre crochets.

Eh bien, nous allons couvrir exactement leur fonction dans la partie suivante, qui risque d'être une jolie lecture plutôt difficile. Donc voici une explication moins lourde qui convient à mon simple cerveau !

Considérons uniquement l'axe X du Microstick. Ce que l'instruction USE MICROSTICK AS MOUSE fait, c'est de couper cet axe en une série de plages ou régions, comme dans le diagramme ci dessous.
(*A titre d'exemple, j'ai fait toutes ces bandes à la même taille, bien que ce ne soit pas exactement le cas avec cette instruction pour les puristes parmi vous.*)



La plage "C" représente le centre de l'axe, ex. le Microstick est dans sa position centrée initiale. Ici bien sur, nous ne voulons pas que la souris bouge. De l'un et l'autre coté du centre, il y a deux plages claires (1), qui représente l'endroit où nous voulons que la souris commence à bouger quand le microstick est déplacé dans ces plages. Et il y a aussi un autre couple de plages (2 & 3) où le microstick se déplacera quand il sera déplacé plus loin de sa position centrée.

Revenons à notre syntaxe :

USE MICROSTICK AS MOUSE (Scale value, Increment value, *optional* Starting value)

La valeur Scale détermine en combien de plages, l'axe du microstick est découpé. La valeur pour le Scale value n'est pas la même que le nombre de plages, c'est réellement une sorte d'équation, mais à la base, au plus la valeur est grande, au plus nombreuses seront les plages qui découpent l'axe.

La valeur "Increment value" détermine à quelle vitesse bouger la souris pour que le microstick se déplace entre chaque plage. Laissez moi expliquer par un exemple, qui illustre ce qui se passe quand le microstick est déplacé de sa position centrale vers une position extrême, comment il bouge entre ces plages. J'en prendrai un pour la valeur de départ plus tard.

USE MICROSTICK AS MOUSE (4, 2)

Plage C: Le stick est dans sa position centrée, et alors la souris ne bouge pas.
Plage 1: La souris commence à bouger à la vitesse donnée par l'Increment value, une "*vitesse de 2*" si vous préférez.
Plage 2: La vitesse de la souris est augmenté par l'Increment value, donc bouge maintenant à une *vitesse de 4*.
Plage 3: La vitesse de la souris est augmenté par l'Increment value, donc bouge maintenant à une *vitesse de 6*.
Plage 4, 5, 6 etc.
Dépends du nombre de plages créé par la Scale value, vous pouvez voir que la souris bouge de plus en plus vite quand le microstick est déplacé entre chaque plage successive. Revenir entre les plages vers la position centrale diminuera aussi bien sur la vitesse de la souris.

Considérons maintenant l'effet de fournir une valeur de départ (starting value).

USE MICROSTICK AS MOUSE (4, 2, 1)

La valeur de départ détermine la vitesse à laquelle la souris bougera dans la plage 1. Apres ça, c'est exactement pareil d'augmenter la vitesse de la souris par l'Incrément value que de bouger le microstick à travers le reste des plages. Donc :

Plage C: Le microstick est dans sa position centrée, et la souris ne bouge pas.
Plage 1: La souris commence à bouger à la vitesse donnée par la valeur de départ, une "*vitesse de 1*" si vous voulez.
Plage 2: La vitesse de la souris est augmenté par l'Increment value, donc bouge maintenant à une *vitesse de 3 (1+2, Starting value + Increment value)*.
Plage 3: La vitesse de la souris est augmenté par l'Increment value, donc bouge maintenant à une *vitesse de 5*.
Plage 4, 5, 6 etc.
Dépends du nombre de plages créé par la Scale value, vous pouvez voir que la souris bouge de plus en plus vite quand le microstick est

déplacé entre chaque plage successive. Revenir entre les plages vers la position centrale diminuera aussi bien sur la vitesse de la souris.

Donc ne vous inquiétez pas si cela ne rentre pas de suite, j'essaie juste de faire passer un message général, le plus grand nombre signifie une souris plus rapide.

Nous pouvons aussi inverser la direction dans laquelle la souris bouge avec le microstick, comme nous l'avons fait avec les autres instructions, nous les avons vu dans un chapitre précédent comme suit :

USE MICROSTICK AS MOUSE (7, 3, 2) - REVERSE_UD
USE MICROSTICK AS MOUSE (7, 3, 2) - REVERSE_LR

Avant de quitter cette section, je devrais pour les besoins de la compréhension parler des autres utilités de cette instruction, car elle attribue la souris aux autres axes.

7.3.1 Assigner les autres axes aux axes de la souris

Maintenant, l'instruction USE MICROSTICK AS MOUSE est en fait un cas particulier de l'instruction suivante (*particulier car cela assigne le bouton gauche de la souris à T1*)

Configuration instructions

Instruction de type 6:

USE *Axis_Identifier* AS *Mouse_Axis* (Scale value, Increment value) - REVERSE_*type*

Pour des instructions Type 5:

USE *Axis_Identifier* AS *Mouse_Axis* (Scale value, Increment value, Starting value) - REVERSE_*type*

ou:

Axis_Identifier est un des axes suivants:

JOYX, JOYY	(tout deux appelles **JOYSTICK**)
THR	
RNG, ANT	(tout deux appelles **ROTARIES**)
MIX, MIY	(tout deux appelles **MICROSTICK**)
LBRK, RBRK	(tout deux appelles **TOEBRAKES**)

RDDR

Mouse_Axis est un des axes suivants:

MOUSE
MOUSEX
MOUSEY
MOUSEZ (la roulette de souris)

Scale value: Un nombre entre 2 et 12. Cela représente le nombre de bandes (graduation) de *Axis_Identifier* .

Increment value: Incrément entre chaque bande, compris entre 1 et 63. *(le 'Scale value' multiplié par' Increment value' doit être inférieures à 128. Ne demandez pas pourquoi!)*

Starting value: la valeur de départ pour une instruction de type 5 à laquelle vous appliquez la valeur incrément. C'est la vitesse initiale de la souris quand elle est bougée depuis sa position centrale et sur son axe *Axis_Identifier* .

REVERSE_*type*: Inverse la direction d'un axe:

REVERSE_UD: quand *Axis_Identifier* se compose de 2 axes (JOYSTICK, ROTARIES, MICROSTICK, TOEBRAKES).

REVERSE_LR: quand *Axis_Identifier* se compose de 2 axes, peut-être utilisée seul ou en conjonction avec REVERSE_UD.

REVERSE_DIR: Inverse un seul axe. Ne peut-être utilisée avec REVERSE_UD, ou REVERSE_LR.

Donc nous pouvons utiliser d'autres axes pour contrôler les axes de la souris, exactement de la même façon. Voici quelques exemples.:

```
USE ROTARIES AS MOUSE (6, 4)
USE JOYSTICK AS MOUSE (11, 2, 0)
USE ANT AS MOUSEY (5, 2) - REVERSE_DIR
USE JOYY AS MOUSEZ (9, 3)
```

La dernière ligne contrôle la roulette de souris en utilisant l'axe Y du joystick!

En résumé, nous avons abordé ici deux instructions qui peuvent être utilisées pour simuler une souris sur le microstick, et dans la dernière section, sur d'autres axes. Rappelez-vous que lorsque nous avons abordé la programmation du HAT, nous pouvions aussi assigner la souris au HAT en utilisant l'instruction:

```
USE HAT1 AS MOUSE (2)
```

En fait, nous pouvons avoir à la fois la souris sur le microstick et la souris sur le HAT, en utilisant le microstick pour bouger la souris rapidement à la position désirée, et le hat pour obtenir de fins ajustements de cette position *[Insérer applaudissement ici]*

Je voudrai expliquer maintenant ce qui se passe ou plutot devrais-je dire comment le compilateur interprete vos instructions et ce qu'il crée comme instructions digitales pour les axes du microstick. C'est assez astucieux à expliquer et pour cette raison je n'ai pas sauté dessus immédiatement. Lme prochaine section est seuleme,nt pour les utilisateurs avances (Croyez moi, cela me demanda de l'experience pour le maitriser!). Mais si vous maitrisez cette partie, alors vous pouvez créer votre propre Souris sur n'importe quel axe, pour avoir les réactions que vous désirez. Vous serez capable de mélangee des instructions souris avec d'autres instructions digitales sur le microstick, ainsi par exemple, le bouton S3 relaché vous permettra de contrôller le Curseur de cible tandis que le bouton S3 appuyé de contrôller votre souris. Astucieux wow !

7.4 Créer une souris personnalisée sur le microstick

Nous avons vu dans la section précèdente comment nous pouvons assigner la souris au microstick avec les instructions USE MTYPE et USE MICROSTICK AS MOUSE. Ceux que font ces instructions tout simplement est la programmation des axes microsticks avec des instructions digitales. Dans cette section, je voudrai vous expliquer comment vous pouvez créer vos propres instructions digitales pour programmer la souris sur le microstick, ou autrepart.

Vous pouvez programmer le microstick avec n'importe quelle instruction digitale, car après tout, ils sont juste des axes avec les mêmes règles que les autres axes. Je vais commencer par une démonstration sur l'utilisation d'instruction de Type 1 et de Type 2, Bien qu'il n'y ai aucune raison de ne pas utiliser l'une des 6 instructions digitales. Jetons un œil sur quelques instructions appropriées de Type 1:

```
MIX 1 14 MSX (2+) MSX (2-) MSX (0)
MIY 1 14 MSY (2-) MSY (2+) MSY (0)
```

Bon, cela ressemble à des instructions de Type 1 ordinaires, excepté que nous avons les signes "-" et "+" d'utilisé. Pour bien comprendre cela, je vais vous accompagner pour voir ce qui arrive quand on bouge le microstick. Rappelons nous commlent une instruction digitale Type1 focntionne. Si nous avons:

```
MIX 1 6 u d c
```

Alors l'axe X, bougé de son extrémité gauche à son extrémité droite puis ramené à son, extrémité gauche produira la séquence de caractères suivante:

u u u c u u u d d d c d d d

Maintenant dans notre exemple, nous avons::

MIX 1 14 MSX (2+) MSX (2-) MSX (0)

Nous ne produisons pas une série de caractères. Les nombres entre les parenthèses specifient quoi ajouter ou quoi soustraire ou buffer courant de la souris – en d'autres mots, à quelle vitesse ou quelle lenteur doir bouger la souris..

Disons que la souris est immobile et que le microstick est en position centrale. La position centrale de l'axe microstick est donnée par le "caractère" centre fonction d'une instruction digitale de l'axe.. MSX (0). Cette instruction informe la souris de garder une vitesse de zéro, d'être immobile le long de son axe X. Comme nous bougeons la souris vers la droite, la souris va commencer à bouger avec un "taux de 2", la valeur 2 est ajouté au buffer de la souris. Bougez le microstick plus vers la droite, et cela bougera la souris à un taux de 4 … bougez le à fond à droite et la souris bougera à un taux de 14 (7*2). Dans le sens inverse, le taux de mouvement de la souris diminue à chaque fois que l'on revient vers le centre du microstick. La souris stoppe alors car une fois encore, nous sommes sur le "caractère" centre de l'instruction. MSX (0).

On observe la même chose quand le microstick est bougé vers le haut ou vers le bas..

Notez que si vous voulez que l'axe Y du microstick bouge de la même facon que la souris (Tirez vers le base et la souris monte), alors l'instruction doit-être:

MIY 1 14 MSY (2+) MSY (2-) MSY (0)

En effet, c'est comme cela car si vous augmentez le buffer Y de la souris, la souris bougera vers le bas de l'écran et vice-versa. Un point important sur la syntaxe est l'utilisation des signes "**+**" et "**-**" entre parenthèses, et le fait qu'ils apparaissent **après** le nombre. Ils indiquent que la valeur placée avant eux doit-être soit ajoutée (+) soit soustrait(-) sur le buffer de l'axe souris. Vous allez comprendre mieux en regardant aux instructions Type 2. J'espère que vous n'êtes pas perdu.*Déjà!*

Je pourrai utiliser aussi ces instructiosn de type 2 pour assigner la souris aux axes du microstick:

MIX 2 9 MSX(-8) MSX(-4) MSX(-2) MSX(-1) MSX(0) MSX(1) MSX(2) MSX(4) MSX(8)
MIY 2 9 MSY(8) MSY(4) MSY(2) MSY(1) MSY(0) MSY(-1) MSY(-2) MSY(-4) MSY(-8)

Ceus sont des instructions digitales type 2. Ils divisent chaque axe en 9 intervalles égales, et assignant les valeurs buffer de la souris à chaque intervalle. Ainsi; bouger le microstick sur son axe X résulte à le bouger intialement à un taux de 1, et le bouger plus augemnetera le taux à 2 puis 4….. et finalement 8. Notons ici la position du signe “-“. Cett fois il est avant le nombre, ce qui signifie qu’il n’est pas soustrait depuis le buffer. En fait, quand le microstick est bougé dans cette intervalle, il prend cette valeur négative.

Au niveau de la syntaxe, notons que les instructions suivantes sont les mêmes:

MIX 2 7 MSX(-4) MSX(-2) MSX(-1) MSX(0) MSX(1) MSX(2) MSX(4)
MIX 2 7 MSX(-4) MSX(-2) MSX(-1) MSX(0) MSX(+1) MSX(+2) MSX(+4)

Si le signe "+" est omis, il est considére comme présent à droite de la valeur. Attendu que:

MIX 2 7 MSX(-4) MSX(-2) MSX(-1) MSX(0) MSX(1) MSX(2) MSX(4)
MIX 2 7 MSX(4-) MSX(2-) MSX(1-) MSX(0) MSX(1+) MSX(2+) MSX(4+)

Pourraient produire des résultats vraiment diffèrents.

Avec tout cela en tête, voyons ce que le compilateur fait quand il voit une instruction USE MICROSTICK AS MOUSE dans notre fichier joystick. Le compilateur convert cette instruction en instruction de type 5 et de Type 6, qui sont respectivement semblables à des instructions de Type2 et de Type 1, excepté que nous pouvons déterminé la taille des intervalles. Rappelons la syntaxe pour cette instruction.:

USE MICROSTICK AS MOUSE (valeur graduation, valeur Increment, valeur de départ *optionnelle*)

J’i dit auparavant que la valeur de graduation ets utilisé pour determiner lenombre d’intervalles de division de l’axe. Effectué par ce calcul:

Nombre d’intervalles = (valeur graduation x 2) - 1

Le compilateur crée alors ces intervalles avec diffèrentes tailles, en utilisant une forumule complexe que je n’expliquerai pas, et crée des instruction Type 6 si aucune valeur de départ est donnée ou des insctructions de Type 5 si une valeur de départ est donnée.

Instructions digitales de Type 6

```
USE MICROSTICK AS MOUSE (2, 2)

MIX 6 3 (2 24 75 98) MSX(2+) MSX(2-) ^
```

MIY 6 3 (2 24 75 98) MSY(2-) MSY(2+) ^
BTN T1 /H MOUSE_LB Rem Bouton gauche de la souris maintenu appuyé quand T1 est pressé.

Maintenant, nous avons effectué ici une chose un peu diffèrente. Je n'ai pas de MSX(0) ou de MSY(0) comme charactère central, comme l'exemple suivant:

MIX 6 3 (2 24 75 98) MSX(2+) MSX(2-) MSX (0)
MIY 6 3 (2 24 75 98) MSY(2-) MSY(2+) MSY(0)

Laissez moi vous expliquer pourquoi il ya des avantages et des inconvénients à avoir des caractères null (^) au lieu de MSX (0), MSY (0). Ces instructions de type 6 diffèrent des instructions Type 5 par le fait qu'elles ajoutent ou soustraient des valeurs aux axes X et Y dans le buffer, tandis que les instructions type 5 régle la valeur du buffer. Quand nous utilisons MSX(0) et MSY(0) comme caractère central, cela reset le buffer souris à zéro comme cela on garanti que la souris arrête de bouger sur la position centrale de l'axe qui la. C'est une bonne chose en général. Mais l'intêret d'utilser des caractères null comme caractère centrak est que nous pouvons assigner ou contrôlller la souris depuis des hats ou des boutons comme nous le faisons avec le microstick et assurer le résultat escoptée. Je peus donc avoir:

USE MICROSTICK AS MOUSE (2, 2)
BTN H1L MSX(1-)
BTN H1R MSX(1+)

Et utiliser alors le microstick pour le contrôle général de la souris et le Hat 1 (gauche –droite) pour un contrôle précis de l'axe X de la souris. C'est digne d'intêret … cela dépend de la manière dont vous voulez que la souris se comporte. Malheureusement il n'y a aucune manière de spécifier au compilateur MSX(0), MSY(0) comme caractère central avec une USE MICROSTICK AS MOUSE. Si vous voulez cela, vous avez besoin d'utiliser les instructions MIX et MIY ou de renseigner la valeur de départ, qui va alors utiliser des instruc tions Type 5 qui MSX(0), MSY(0) comme caractère. Voyons ce que le compilateur fait avec plusieurs instructions Type 6 USE MICROSTICK AS.

USE MICROSTICK AS MOUSE (3, 4)

est converti en:

MIX 6 5 (2 16 32 68 84 98) MSX(4+) MSX(4-) ^
MIY 6 5 (2 16 32 68 84 98) MSY(4-) MSY(4+) ^
BTN T1 /H MOUSE_LB

USE MICROSTICK AS MOUSE (4, 2)

```
est converti en:

MIX 6 7 (2 12 23 36 65 78 89 98) MSX(2+) MSX(2-) ^
MIY 6 7 (2 12 23 36 65 78 89 98) MSY(2-) MSY(2+) ^
BTN T1 /H MOUSE_LB
```

```
USE MICROSTICK AS MOUSE (12, 3)

est converti en:

MIX 6 23 (2 4 6 8 11 14 17 21 25 30 36 43 58 65 71 76 80 84 87 90 93 95 97 98) MSX(3+) MSX(3-) ^
MIY 6 23 (2 4 6 8 11 14 17 21 25 30 36 43 58 65 71 76 80 84 87 90 93 95 97 98) MSY(3-) MSY(3+) ^
BTN T1 /H MOUSE_LB
```

```
USE MICROSTICK AS MOUSE (4, 2) - REVERSE_UD

est converti en:

MIX 6 7 (2 12 23 36 65 78 89 98) MSX(2+) MSX(2-) ^
MIY 6 7 (2 12 23 36 65 78 89 98) MSY(2+) MSY(2-) ^
BTN T1 /H MOUSE_LB
```

Maintenant regardons ce qui arrive quand nous renseignons la valeur de départ lors de l'utilisation de l'instruction USE MICROSTICK AS MOUSE . le compilateur les converti en instructions de Type 5.

Instructions digitales de Type 5

```
USE MICROSTICK AS MOUSE (2, 2, 3)

est converti en:

MIX 5 3 (0 24 75 100) MSX(-3) MSX(0) MSX(3)
MIY 5 3 (0 24 75 100) MSY(3) MSY(0) MSY(-3)
BTN T1 /H MOUSE_LB
```

Notons que le nombre d'intervalles créées est de 3. L'intervalle centrale est utilisé pour assurer que la souris est immobile, et les 2 bandes de chaque coté augmentent la valeur de départ, ainsi effectivement, la veleur incrément est ignorée dans ce cas. Maintenant comparons cette instruction avec la suivante ci-dessous:

```
USE MICROSTICK AS MOUSE (3, 2, 3)

est converti en:
```

```
MIX 5 5 (0 14 31 69 86 100) MSX(-5) MSX(-3) MSX(0) MSX(3) MSX(5)
MIY 5 5 (0 14 31 69 86 100) MSY(5) MSY(3) MSY(0) MSY(-3) MSY(-5)
BTN T1 /H MOUSE_LB
```

Maintenant vous pouvez voir la relation entre la valeur incrément et la valeur de départ. Et si vous ne pouvez pas, comparez l'instruction suivante et ce que vous devriez avoir alors.

```
USE MICROSTICK AS MOUSE (3, 8, 1)
```

est converti en:

```
MIX 5 5 (0 14 31 69 86 100) MSX(-9) MSX(-1) MSX(0) MSX(1) MSX(9)
MIY 5 5 (0 14 31 69 86 100) MSY(9) MSY(1) MSY(0) MSY(-1) MSY(-9)
BTN T1 /H MOUSE_LB
```

```
USE MICROSTICK AS MOUSE (12, 2, 3)
```

est converti en:

```
MIX 5 23 (0 2 4 6 9 12 16 20 25 30 36 43 59 66 72 77 82 86 90 93 96 98 99 100)
        MSX(-23) MSX(-21) … MSX(-3) MSX(0) MSX(3) … MSX(21) MSX(23)
MIY 5 23 (0 2 4 6 9 12 16 20 25 30 36 43 59 66 72 77 82 86 90 93 96 98 99 100)
        MSY(23) MSY(21) … MSY(3) MSY(0) MSY(-3) … MSY(-21) MSY(-23)
BTN T1 /H MOUSE_LB
```

```
USE MICROSTICK AS MOUSE (3, 8, 1) - REVERSE_UD, REVERSE_LR
```

est converti en:

```
MIX 5 5 (0 14 31 69 86 100) MSX(9) MSX(1) MSX(0) MSX(-1) MSX(-9)
MIY 5 5 (0 14 31 69 86 100) MSY(-9) MSY(-1) MSY(0) MSY(1) MSY(9)
BTN T1 /H MOUSE_LB
```

Et dans un soucis de perfection, l'assignement des autres axes:

```
USE JOYSTICK AS MOUSE (3, 2, 3)

JOYX 5 5 (0 14 31 69 86 100) MSX(-5) MSX(-3) MSX(0) MSX(3) MSX(5)
JOYY 5 5 (0 14 31 69 86 100) MSY(5) MSY(3) MSY(0) MSY(-3) MSY(-5)
```

```
USE JOYY AS MOUSEZ (4,2)
```

est converti en:

JOYY 6 7 (2 12 23 36 65 78 89 98) MSY(2-) MSY(2+) ^

Et bien espérons que j'ai réussi a expliquer cela clairement ! Continuons.

7.5 USE ZERO_MOUSE

cette instruction de configuraion est utilse quand vous utilisez vos propres instructions souris avec le microstick, en conjugaison avec les attributs /I et /O modifiers, pour éviter à la souris de se coincer.

Nous avons vu dans la section précèdente commenet nous pouvons utilisez des instructions digitales pour créer une souris personnalisée sur le microstick. Une insctruction de configuration USE ZERO_MOUSE pour s'assurer que vous n'obtiendrez pas une souris coincée quand vous cominerez les instructions souris avec les attributs /I et /O . Considérons cet exemple:

```
MIX   /I 1 6 MSX(2+) MSX(2-)
      /O 1 6 RARROW LARROW
MIY   /I 1 6 MSY(2-) MSY(2+)
      /O 1 6 UARROW DARROW
```

dans cet exemple, si vous maintenez appuyé le bouton S3 de votre joystick, le microstick bougera alors la souris. Si tandi que la souris bouge, vous relachez le bouton S3, la souris continuera à bouger et elle sera coincée quand elle atteindra l'un des bords de l'écran. Insérer une instruction USE ZERO_MOUSE évitera que cela, forçant la souris à s'arrêter quand le bouton S3 est appuyé/relaché.

C'est très utilse d'inster la dessu, car cette sours coincée n'est pas un bug en tant que tel. Le control le la souris par le microstick se comporte comme il a été programmé pour se comporter. Vous pouvez toujours eviter qu'une souris se coince en amenant le microstick à sa position centrale avant de relacher le bouton S3 avec l'exemple cité plus haut. Cette règle d'or est valable ici quand les touches ou la souris se coincent : Vous devez utiliser les boutons de votre controlleur, les hats et le saxes comme si ils avaient été conçus pour être utiliser à travers de la programmation. Cette recommendation est un superbe travail pour ceux d'entre nous qui ne suivent jamais cette rêgle ☺.

7.6 programmer les boutons souris

Vous pouvez assigner les boutons de la souris avec des instructions bouton souris, instructions axes… toutes sortes d'instructions. Voici la syntaxe pour les utiliser:

Syntaxe des boutons souris:

MOUSE_LB
MOUSE_RB
MOUSE_MB

(MB = Bouton du milieu sur une souris à 3 boutons)

Voici un exemple:

```
BTN T1 /I /H MOUSE_RB
       /O /H MOUSE_LB
```

Avec cette instruction, nous assignons le bouton gauche de la souris au bouton T1 du microstick quand le bouton S3 n'est pas appuyée, et le bouton droit de la souris au bouton T1 du microstick quand le bouton S3 est appuyé.

Vous pouvez aussi KD et KU avec les boutons souris:

```
BTN S1 KD(MOUSE_LB) DLY(2000) KU (MOUSE_LB)
```

Quand le bouton S1 est pressé, le bouton gauche de la souris est pressé pour 2 secondes, puis relaché..

7.7 Désactiver l'assignement par défaut de la souris au microstick

Dans la fenêtre des préférences de l'application Foxy, il y un onglet intitulé "Defaults." L'objectif de ces paramètres est d'avertir le compilateur d'utiliser ces paramètres si aucune instruction contraire n'existe dans le fichier.

L'un de ces paramètres est "Assign mouse to microstick." Si c'est séléctionné, alors quand vous chargerez in fichier dans votre controlleir, le compilateur parametrera la souris sur le microstick et le bouton gauche de la souris sur T1. c'est prévu de cette manière car la plupart des utilisateurs veulent que le microstick contrôlle la souris par défaut mais n'ont pas pour autant été plus loin dans le manuel de référence pour comprendre quelle instruction permet de le faire!

Si vous voulez que les axes du microstick soient assignables sous un jeu, mais sans avoir la souris assignée par défaut, sans avoir à désélectionner l'option par défaut dans Foxy, alors vous devrez utiliser l'Déclaration de configuration::

Déclaration de configuration:

DISABLE MOUSE

dans votre fichier joystick. Cela empêchera le compilateur d'assigner l'instruction digitale par défaut au microstick *si vous n'avez pas paramétré les préférences de Foxy pour le forcer à le faire*.

Vous pouvez toujours assigner la souris à un hat, a des boutons ou a d'autres axes. Rappelez vous que le périphérique souris est toujours présent – c'est juste que vous devez l'assigner à autre chose si vous désirez utiliser la souris..

7.8 Instructions mouvements avancés souris

Nous avons déja vu la possibilité de bouger la souris à travers des instructions de programmation. Dans la dernière partie de ce chapitre, nous allons nous attarder sur les possibilités de mouvement complexe de la souris.

7.8.1 Définir la résolution d'écran

Avec toutes les instructions qui vont bientôt être expliquées, il est essentiel de définir la résolution d'écran que vous utilisez dans votre jeu, avec une Déclaration de configuration::

Déclaration de configuration::

USE SCREEN_RESOLUTION (X,Y)

Exemple:

USE SCREEN_RESOLUTION (800,600)

Les valeurs les plus basses que vous pouvez utiliser sont respectivement 640, 480 pour X et Y.

Note technique: *800 et 600 décrivent la résolution d"écran an pixels. Un pixel est le plus petit point que vous pouvez dessiner à l'écran. Ainsi dans cet exemple, cela consiste en un écran de 800 lignes et de 600 colonnes de points. Evidemment à une résolution d'écran de 1600 par 1200, votre résolution d'écran est pls haute, signifiant que l'image sera pluis détaillée.*

Nous pouvond donc utiliser les instructions suivantes avec la souris:

Syntaxe de la commande:

Aller à une position spécifique de l'écran
MOUSEXY (*Origin*,X,Y)

Mouvement relatif à la position courante de la souris
MOUSEMOVE (X,Y)

Mouvement Circulaire/Polygonal
MOUSEROTATE (*Origin*, CentrePoint, Radius, Start angle, Macro1, Rotate direction, Final angle, Number of steps, Macro2)

7.8.2 Aller à une position spécifique de l'écran

Syntaxe de la commande:

MOUSEXY (*Origin*,X,Y)
MOUSEXY (*Origin*,X%,Y%)

avec

- *Origin* est l'un des coins de l'écran UL, DL, UR, DR.
- X,Y sont les coordonnées X, Y sur lécran ou vous voulez que la souris aille.
- X%, Y% est un pourcentage de mouvement de souris compris entre 0 et 100% (Exactitude à 3 décimales près). Utiliser des pourcentages permet de changer de résolution d'écran sans nécéssairement changer les valeurs des instructions.

Il n'y a aucune possibilité dans un jeu de connaître la position de la souris à un instant donné ou de la suivre. C'est pourquoi, dans le but de bouger vers une certaine position, nous devons auparavant bouger la souris à un des coins de l'écran, ainsi nous connaissons exactement les corrdonnées de cette position depuis l'instruction SCREEN_RESOLUTION, et le compilateur peut calculer ou placer la souris. Cela arrive tellement peu souvent que ca ne devrait pas poser problème., mais vou devez vous assurer qu'aucun bouton de souris n'est pressé avant de faire cela.

Ainsi l'instruction:

BTN H3D MOUSEXY (UL, 400, 300)

Bougera d'abord la souris très rapidement vers le coin en haut à gauche de l'écran, puis la placera au point 300,100, ui est le centre de l'écran quand la résolution est en 800 âr 600. Disons que vous avez besoin de presser F2 pour afficher la vue cockpit, puis de bouger la souris sur un bouton du cockpit, de

presser le bouton, et de retourner à la vue frontale en pressant F1. Cette instruction le fera:

BTN H3D F2 MOUSEXY (UL, 400, 300) MOUSE_LB F1

Maintenant si nous changeons cette instruction comme cela:

BTN H3D F2 MOUSEXY (UL, 50%, 50%) MOUSE_LB F1

si un utilisateur change alors la résolution de son jeu en 1600 par 1200, et change l'instruction USE SCREEN_RESOLUTION pour refleter ces valeurs, alors l'instruction fonctionnera comme en 800 par 600.

NOTES

1. *Vous devez avoir une Déclaration de configuration: USE SCREEN_RESOLUTION ( ) présente pour utiliser cette instruction.*

2. *Toutes les instructions souris sont sujettes aux restrctions du taux de vitesse. Si le mouvement de la souris est trop lent alor reduizez la valeur de l'instruction RATE. Notez que par défaut, si vous n'avez pas d'instruction USE RATE (nnn) dans votre fichier, il sera paramétré à 0 – taux de réponse le plus rapide de touta façon.*

7.8.3 Mouvement relatif à la position courante de la souris

Syntaxe de la commande:

MOUSEMOVE (X,Y)
MOUSEMOVE (X%,Y%)

where:

- X,Y représente le nombre de pixels de déplacement de la souris depuis sa position courante.
- X%, Y% est un pourcentage de mouvement de souris compris entre 0 et 100% (Exactitude à 3 décimales près).. Utiliser des pourcentages permet de changer de résolution d'écran sans nécéssairement changer les valeurs des instructions..

Généralement vous saurez ou la souris est parce que vous utiliserez l'instruction a MOUSEXY pour positionner la souris au départ.

Regardons quelques exemples utilisant l'instruction MOUSEMOVE :

BTN TG1 MOUSEMOVE (20, 50)

Appuyer sur la gachette bougera la souris de 20 pixels vers la droite le long de l'axe horizontal de l'écran, et de 50 pixels vers le bas le long de l'axe vertical de l'écran.

BTN S1 MOUSEMOVE (20%, 50%)

Appuyer sur le bouton S1 bougera la souris sur 20% de la largeur de l'écran le long de l'axe horizontal, et de 50% de la hauteur de l'écran le long de l'axe vertical. Donc si vous utilisez du 800 par 600, cela signifie que la souris bougera de 160 pixels vers la droite (20% de 800) et de 300 pixels vers le bas (50% de 600).

BTN H2R MOUSEMOVE (30, 0)

Appuyer vers le haut le Hat 2 bougera seulement la souris de 30 pixels vers la doite selon une ligen horizontale. Si vous vouliez bouger la souris à gauche, vous auriez du placer un signe '-' avant le 30.

BTN H2U MOUSEMOVE (0%, -50%)

Appuyer vers le haut le Hat 2 up bougera la souris de 50% de la hauteur de l'écran – équivalent à 300 pixels pour une résolution d'écrab de 800 par 600..

NOTES

1. *Vous devez avoir une Déclaration de configuration: USE SCREEN_RESOLUTION ( ) présente pour utiliser cette instruction, ou il y aure une erreur de compilation.*

2. *Toutes les instructions souris sont sujettes aux restrctions du taux de vitesse. Si le mouvement de la souris est trop lent alor reduizez la valeur de l'instruction RATE. Notez que par défaut, si vous n'avez pas d'instruction USE RATE (nnn) dans votre fichier, il sera paramétré à 0 – taux de réponse le plus rapide de touta façon.*

3. Vous ne pouvez pas utiliser en même temps des valeurs absolues et des valeurs relatives dans ces instructions*:*

BTN H2U MOUSEMOVE (0%, -50%) *est correcte, attentu que:*

BTN H2U MOUSEMOVE (0, -50%)

Générera une erreur de compilation.

4. Avec une instruction MOUSEMOVE *, la souris bougera le long des deux axes au même moment. Donc une instruction comme:*

BTN S1 MOUSEMOVE (100, 100)

Bougera la souris diagonalement à droite vers le bas, et non pas 100 pixels le long de l'axe X puis 100 pixels le long de l'axe Y.

7.8.4 Mouvement Circulaire/Polygonal

Syntaxe de la commande:

MOUSEROTATE (Origine, PointCentre, Rayon, angle départ, [Macro], Direction de rotation, Angle final, Nombre de pas)

Avec :

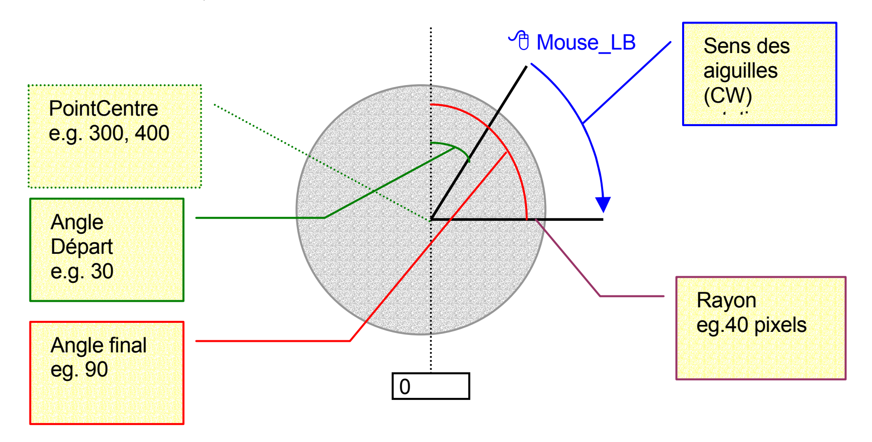
- *Origine* est un des coins de l'écran UL, DL, UR, DR.
- *PointCentre* est le centre de rotation représenté par des coordonnée X, Y ou par des %
- *Rayon* est le rayon d'un cercle pour définir l'arc de rotation en pixels ou en pourcentage (*Voir les notes de la section*)
- *Angle départ* va de 0 à 360°
- *Macro* est une simple macro … typiquement l'appui sur le bouton de la souris (*Voir les restrictions dans les notes de la section*.) utiliser un caractère null si vous ne voulez pas insérer une macro. La macro doit être comprise entre des parenthèses crochets [ ].
- *Direction de rotation* est soit CW soit CCW (Sens des aiguilles d'une montre ou sens inverse)
- *Angle final* représente l'angle à atteindre entre 0 et 1800° (5 Révolutions complètes)
- *Nombre de pas* défini la fluidité de la rotaition. Les valeurs sont 1 ou 2. Ces valeurs produisent des mouvements de diffèrentes formes:

Nombre de pas = 1: un carré (*err.. 4 cotés!*),
Nombre de pas = 2: an octogone (*8 cotés*)

Plus le pas sera important, plus lent sera le mouvement, bien que ce ne soit pas dramatique. *Toutes les valeurs numériques sont précises à une décimale près (244.3, 301.8)*

Cette fonctionnalité fut implémentée pour permettre à l'utilisateur de tourner un bouton dans un cockpit, juste en pressant un bouton du controlleur.

Prenons un exemple, en appiuyant sur le bouton gauche de ma souris, je veux tourner les aiguilles de communication de mon cockpit dont le centre est loacalisé sur l'écran à 300,400.



Les instructions dont j'ai besoin sont:

USE SCREEN_RESOLUTION (1024,768)
BTN S2 MOUSEROTATE (DL, 300, 400, 40, 30, [MOUSE_LB], CW, 90, 2)

Voyons ce qui se passe par étape. Quand le bouton S2 est pressé:

1. DL – la souris bouge vers le coin en bas à gauche de l'écran très rapidement pour obtenir un point de référence. Rappelons que la souris peut-être située n'importe ou à l'écran, donc nous devons la bouger à un coin ou au centre pour obtenir une position de référence.
2. 300,400 – La souris utilise le centre des aiguilles de comunications de l'écran pour calculer ou est le centre de rotation.
3. 40 – le rayon de l'arc de rotation.
4. 30 – l'angle de départ depuis la verticale (0 degrés), combiné avec les informations précédentes determine ou la macro1 sera activée..
5. [MOUSE_LB] – le bouton gauche de la souris est maintenue enfoncée.
6. CW – La rotation s'effectue dans les sens des aiguilles d'une montre
7. 90 – jusqu'à ce que la rotation atteigne 90 degrès depuis la verticale (0 degré)
8. 2 – le mouvement va suivre la forme d'un octogone. Utilisez le plus petit nombre pour que les aiguilles de communications atteignent la valeur désirée.
9. L'instruction est terminée donc la macro se termine, le bouton gauche de la souris est relachée.

C'est complexe, je le sais, mais décrire le chemin que suis la rotatation de la souris l'est tout autant! :-)

NOTES SUR LA MACRO

1. *La macro sera toujours maintenus – c'est comme utilisé l'attribut /H devant(maintenir). Le code pour la macro sera automatiquement interrompue uen fois que le mouvement de la souris est terminée.*

2. Voici la liste des touches 'pressées' autorisées....

- *Caractères* (a, b, 1, 2, `, *etc.*)
- *Caractères* SHF, ALT, CTL (A, ALT b, *etc.*)
- *Touches DirectX et POV* (DX1, POVL, *etc.*)
- Touches MOUSE_LB / RB / MB
- Touches XFlag (X1, X2, etc.)
- Touches USB

3. Vous ne pouvez pas utiliser des attributs '/' DLY( ) *or* RPT( ) *dans la Macro.*

NOTES

1. Vous ne pouvez pas grouper des instructions MOUSEROTATE avec des séquences de touches. *ainsi:*

BTN S2 a b DLY(30) MOUSEROTATE (*blah blah)* PRNTSCRN

est bon:

BTN S2 a b DLY(30) **{**MOUSEROTATE (*blah blah*) PRNTSCRN**}**

Générera une erreur de compilation.

2. *Vous devez avoir une Déclaration de configuration: USE SCREEN_RESOLUTION ( ) présente pour utiliser cette instruction, ou il y aure une erreur de compilation.*

3. *Toutes les instructions souris sont sujettes aux restrctions du taux de vitesse. Si le mouvement de la souris est trop lent alor reduizez la valeur de l'instruction RATE. Notez que par défaut, si vous n'avez pas d'instruction USE RATE (nnn) dans votre fichier, il sera paramétré à 0 – taux de réponse le plus rapide de touta façon. .*

4. *Pour la macro, vous pouvez utiliser des macros de fénitions plutot que de programmer directement. Les macros les plus rencontrés* MOUSE_LB ou MOUSE_RB (*respectivement les boutons gauche et droite de la souris.*)

5. *Les touches combinées ne fonctionneront pas dans l'instruction* MOUSEROTATE *. Comme tout dans la macro est groupé pour la rotation de la souris, vou devez utiliser* LSHF *et non pas* SHF.

6. *Le rayon peut aussi être exprimé en pixels ou en % de la résolution d'écran de l'axe. cela semble bizarre au départ donc laissez moi vous l'expliquer. le rayon est une longueur fixe normalement exprimé en pixels. Disons que nous avons une résolution d'écran de 800 par 600, et que nous avons un rayon de 600 pixels (le centre de la rotation etant lié le long du bas de l'axe X). Improbable, mais ca peut arriver. Si le rayon est exprimé en % (600/800*100) = 75%. Si vous voulez désormais utiliser une résolutioon diffèrente de 1024*768, le rayon sera de 75% de 1024, soit 768 pixels de long. Comme il est possible d'avour un rayon plus long que la résolution Y de l'écran, alors cela a un sens de l'exprimer en pourcentage de l'axe X. Il y a un point important à aborder ici. Le ratio de l'axe X par rapport à Y est critique ici. Maintenant pour les résolutions de 800 par 600, 1024 par 768, 1152 par 864, 1600 par 1200 etc. le ratio est constan et égal à 1.333. mais si vous utilisez une résolution comme 1280 par 1024 (ratio = 1.25) alors comme le ratio est diffèrent, le rayon ne sera pas retaillée précisemment. Donc n'oubliez jamais cela si vous paramétrez l'instruction* MOUSEROTATE *en utilisant des % plutôt sur des mesures en pixels.*

7. C*'est une instruction cmoplexe, et donc les chances de provoquer des erreurs de compilation si vous oubliez une des parties, si vous placez mal une virgule sont très importantes! Je vous supplie d'utiliser l'assistant avancé de programmation de la souris dans Foxy pour créer ces instructions. Au moins c'est une manière de coder très astucieuse et je l'apprécierai si cela n'avait pas été en vain!*

8. Programmation logique

8.1 Programmation logique – les bases

8.1.1 Comprendre les flags

la programmation logique est entièrement basé su le concepte de **flags**. Alors qu'est-ce qu'un flag? Commençons avec des faits concernants les flags qui vont totalement vous mettre dans la confusion la plus totale, et nous les rendrons très clairs avec un exemple simple. Voici les faits:

- Un flag peut-être on ou off.
- Il y a 32 flags, et ils sont nommés de X1 à X32.

- Un flag ne fait rien, il est juste on ou off.

Confus? (*ce fut un enfer quand je pris au départ connaissance de ces flags*) Utilisons donc comme example le bouton clavier Caps Lock pour expliquer le concepte des flags.

Disons que votre bouton Caps Lock se nomme X1. Quand vous pressez la touche Caps Lock, la lumière de la touche s'allume et donc X1 devient on. Le clavier sait que X1 est on et envoi donc un caractère "W" au lieu d'un caractère "w" quand vous pressez la touche "w". Si je presse le bouton Caps Lock pour le rendre off, la lumière de la touche s'eteind, et maintenant un caractère "W" est envoye à l'ordinateur. Nous pouvons docndire que la touche Caps Lock ne fait *rien du tout* actuellement. C'est juste un switch X1 on ou off, mais c'est le fait que X1 soit on ou off qui conditionne quel caractère sera envoyé à l'ordinateur par le clavier..

C'est donc un cocnept important à. Un flage ne fait rien du tout. Il est juste on ou off. Vous ne pouvez pas voir qu'il est on ou off mais vous pouvez le fait qu'il soit on ou off pour changer le comportement de votre joystick. Ce que nous avons à faire dans la programmation logique est de programmer ce qui doit arriver en fonction que le flag soit on ou off.

8.2 Définir les flags logiques et leurs instructions bouton

Commençons par comprendre comment rendre on ou off un flag, voici la syntaxe:

Déclaration de configuration::

```
DEF X1 S2
```

Instruction de programmation logique:

```
BTN X1 /H Fire_rockets
```

Dans l'exemple ci-dessus, nous avondsDEFini un flag appelé X1 en utilisant l'instruction DEF, et nous avons décidé qu'il serait controllé par le bouton S2. Quand le bouton S2 du joystick est appuyé, le flag X1 est on et quand le bouton S2 est relaché, le flag est alors off.

Une fois qu'un flag logique est défini, vous pouvez le programmer avec l'instruction bouton (BTN). Et dans l'exemple ci-dessus, quand le bouton S2 est appuyé, des roquettes seront continuellement tirées. Elle sont tirées parce que vous avez demandé ces tirs lorsque X1 est on, et il reste on tant que le bouton S2 est préssé. C'est important de noter que les instructions de bouton logique

sont assujetis aux même règles que les instructions de boutons ordinaires excepté le fait que vous ne pouvez pas utiliser l'attribut /T avec eux. Ils se comporteront donc comme des instructions non répétées, même si le flag est on, à moins que vous utilisiez quelque chose comme l'attribut /H qui permet de maintenir le bouton logique enfoncé..

Ok, vous êtes probablement entrain de vous frotter la tête en vous posant la question: '*Bien, pourquoi ne faisons nous pas simplement :*"

BTN S2 /H Fire_rockets

dans notre fichier? Pourquoi avoir besoin de ce flag logique?" Manifestement dans cet exemple, vous avez absolumment raison, nous n'avons pas besoin de ces flags logiques et les 2 instructions sont équivalentes. Mais nous allons aborder bientôt des exemples que vous ne pourrez pas résoudre avec une programmation classique. – Pour le moment je présente seulement la syntaxe.

Il est aussi possible de définir des flags logigues directement dur des instructions de Type digitale et directement avec des instructions boutons:

RNG 2 5 X1 X2 X3 X4 X5
BTN H1L X8

Sont des instructions parfaitement valables.

Maintenant, il y a une légère diffèrence entre definir un flag logique utilisant une Déclaration de configuration: et le définir directement sur un bouton. Quand vous utilisez quelquechose comme cela:

DEF X20 S1

alors le flag X20 est on et reste on tant que S1 est pressé. Maintenant si nous avons:

BTN S1 X20

dans notre fichier joystick (*sans instruction DEF*) alors cela va se comporter comme une instruction normale de bouton, X20 passera on puis off même si vous restez appuyé sur le bouton S1. Donc

BTN S1 /H X20

va se comporter de la même façon, qu'une instruction DEF X20 S1 . Notons que nous ne pouvons pas avoir une instruction DEF qui défini un flag logique dans le but de l'utiliser sur une instruction bouton ou sur une instruction axe.

Nous venons de dire que placer un flag logique sur une instruction bouton est sujet à la même programmabilité que les instructions normales. Donc vous pouvez generer des fliags avec cette instruction:

BTN S1 KD(X8) DLY(2000) KD(X6) KU(X6 X8)

Cele ne semble pas présenter un avantage particulier, mais si vous vouliez une étape de paramétrage suivi par une étape répétitive, vous pourriez faire cela:

BTN X1 /A Fire_Main_Guns
BTN S1 /H Switch_to_Main_Guns X1

Maintenant /H est utilisé et cela s'applique sur l'instructions concernant X1. La conséquence de tout cela est que Switch_to_Main_Guns est interprété une seule fois, mais que Fire_Main_Guns est auto-répété par le flag logique. Cool, non?

NOTES

Faites attention lorsque vous définissez le même flag logique à la fois sur une instruction DEF et sur une instructions. Par exemple:

DEF S4 X1
BTN S2 X1

C'est effectivement la même chose que:

DEF X1 S4 OR S2

donc si S4 ou S2 sont pressés, X1 sera on.

8.3 Opérateurs logiques

Les opérateurs logiques sont les suivants : AND, NOT et OR qui peuvent aussi être utilisés en conjonction avec des parenthèses.

Instructions de Configuration:

DEF X1 S2 OR T6
DEF X2 S4 AND S3 AND H1U
DEF X3 S1 AND NOT X1

Instructions de programmation logique:

BTN X1 Fire_missile
BTN X2 Engines_off Gather_belongings Eject
BTN X3 AutoPilot

Regardons ces deux lignes:

```
DEF X1 S2 OR T6
BTN X1 Fire_missile
```

Le flag X1 peut être on en appuyant sur le bouton S2 ou le bouton T6. L'utilisation de S2 ou de T6 entraine donc le même résultat, un missile est tiré. Nous aurions pu avoir le même résultat avec:

```
BTN S2 Fire_missile
BTN T6 Fire_missile
```

Regardons un cas qui ne peut être programmé directement sur les boutons:

```
DEF X2 S4 AND S3 AND H1U
BTN X2 Engines_off Gather_belongings Eject
```

Dans cet exemple, le flag X2 devient on seulement quand le bouton S4 et le bouton S3 sont pressés et que le Hat est en même temps poussé vers le haut. C'est une chose que vous ne pouvez pas faire! Mais quand vous le ferez, vous atterirai sur votre postérieur. Et pour finir, un autre exemple:

```
DEF X1 S2 OR T6
DEF X3 S1 AND NOT X1
BTN X3 AutoPilot
```

Nous avon définis que X3 sera tourné on par S1, mais pas dans le cas ou le bouton S2 ou T6 est préssé. Donc S1 tournera on l'autopilot a moins que S2 ou T6 soit pressé.

Notons que les flags logiques suivent les mêmes comportements non répétitifs que les instructions bouton quand ils sont programmés directement sur des instructions boutons.
Avant de terminer les opérateurs logiques, notons qu'il est possible d'utiliser des parenthèses pour grouper des instructions logiques ensemble. L'instruction DEF peut-être combiné avec des instructions AND, OR, et NOT, des parenthèses droites et gauches, et des boutons et flag, pour donner une équation logique, par exemple:

```
DEF X1  (S1 AND NOT S2) OR (X5 AND (H1U OR H2U))
```

est parfaitement valide.

8.4 La bascule logique

Dans les examples que nous avons utilisé, nous avons généré des flags logiques qui passent on quand un ou une combinaison de bouton est pressé, et le flag passe alors off quand ces boutons sont relachés. Il est aussi possible de forcer un flag logique à se comporter un peu à la manière d'un interrupteur de lumière.. de rester on même quand le bouton est relaché, et de passer off quand le bouton est pressé une nouvelle fois. En d'autres mots, nous allons basculer entre les etats on et off du flag. Nous pouvons le faire en mettant un "*" après le flag ou le bouton.

Déclaration de configuration::

DEF X1 S4*

Instruction de programmation logique:

BTN X1 /A Chaff DLY(30) Flare DLY(30)

Quand le bouton S4 est pressé, le flag X1 devient on. Il reste on même si S4 est relaché, grâce à l'utilisation de la bascule *. Le flag deviendra off seulement quand le bouton S4 sera pressé de nouveau. Les conséquences de l'exemple ci-dessus vous permettent en pressant /relachant S4 de commencer à lancer des contre-mesures (chaff,flares) depuis votre avion tout en évitant les SAM et les sidewinder (*Jour malchanceux*) et d'appuyer une nouvelle fois sur S4 pour stopper le lancement des chaff et flare. (Si vous en aviez encore!)

NOTES

1. *Vous ne pouvez pas référencer une bascule logique directement sur un bouton. Par exemple:*

BTN T6 X1*

Cela générera une erreur de compilation. (ce comportement est diffèrent de la syntaxe logique TM F-22 PRO). Les bascules logiques sont uniquement autorisées sur les instructions DEF. Cela devrait fonctionner correctement:

DEF X1 T6*

2. *Notons qu'à la diifèrence de l'instruction bouton, l'attribut /T n'est pas autorisé avec des instructions boutons de programmation logique. donc:*

BTN X7 /T a /T b

Cela générera une erreur de compilation. Par contre ceci est autorisé:

BTN S4 /T X1 /T X2 /T X3

8.5 Utiliser les fonctions logiques DELAY et PULSE

8.5.1 La fonction Delay

La fonction DELAY ajoute un temps fixe entre le moment ou l'equation logique retourne un etat VRAI et le moment ou le flag pass on. La syntaxe est:

Déclaration de configuration::

```
DEF Xflag  DELAY(Flag_on_delay) Logical equation
```

Considérons cet exemple:

```
DEF X1 DELAY(1000) S1 AND S4
BTN X1 Eject
```

Cette instruction défini que X1 passera on 1 seconde (1000 millisecondes) **après** que S1 et S4 soient pressés silmutanément. Si S1 ou S2 est relaché, le delai est stoppé. Si X1 n'est pas encore passé ON, il restera OFF, si il est passé ON, il re-passera à OFF immédiatement. Le délai sera aussi remis à zéro, et donc si S1 et S4 sont préssés de nouveau, le délai de 1 seconde re-démarrera une nouvelle fois.

Et dans cet exemple, c'est utilisé dans le but d'une séquence d'éjection.

NOTES

1. *La fonction logique DELAY est totallement diffèrente de la focntion DLY(). Les valeurs utilisés avec l'instruction DELAY sont comprises entre 0 et 327670 (soit 5.46 minuytes, ne me demandez pas pourquoi !).*

2. L'instruction DELAY doit apparaître en premier dans une instruction logique. Donc :

DEF X1 DELAY(1000) S1 AND S4 *est OK, mais:*

DEF X1 X2 AND DELAY(1000) S1 AND S4 générera une erreur.

8.5.2 La fonction Pulse

La fonction PULSE envoi des séquences répétés ON-OFF tant que l'équation logique est vraie. Les 2 nombres dans la focntion definissent respectivement les deux périodes ON et OFF. Voici la syntaxe:

Déclaration de configuration::

```
DEF Xflag  PULSE(Time_on Time_off) Logical equation
```

En exemple:

```
DEF X1  PULSE(100 1000) H1U
BTN X1 Trim_up_increase
```

Presser le HAT 1 fera passer X1 ON immédiatement pendant 1/10ème de seconde, puis OFF pendant 1 seconde, puis ON pendant 1/ème de seconde, etc,etc. Cette opération continuera tant que le HAT 1 sera préssé. Et dans cet exemple, nous avons augmenté le trim de 1 niveau toutes les 1.1 secondes. Notons que c'est bien 1.1 secondel. Si je voulais aumenter le trim toutes les secondes, l'instruction devrait être changée comme cela ::

```
DEF X1  PULSE(100 900) H1U
```

Notons que le caractère ESPACE doit séparer les deux valeurs numériques. Utiliser une virgule ou un autre séparateur entrainera une erreur de compilation.

NOTES

1. L'instruction PULSE doit apparaître d'abord dansnune instruction logique. *Donc:*

DEF X1 PULSE(100 1000) S1 AND TG1	*est OK, mais*
DEF X1 X2 AND PULSE(100 1000) S1	générera une erreur.

2. *La précison actuelle pour un temps DELAY, RATE, PULSE est environ 30 millisecondes, et la valeur minimum est aussi de 30, donc n'importe quelles valeur entre 1 et 30 produit un delai de 30 millisecondes, 31 à 60 produit un délai de 60 millisecondes, 61 à 90 produit un délai de 90 millisecondes etc, etc.*

3. *Les valeurs autorisées pour l'instruction logique* PULSE *sont comprises entre 0 et 82800000 (en exemple 23 heures).*

8.5 Exemples de programmation logique

8.5.1 Alterner une instruction de type 4 entre ON et OFF

En utilisant une instruction digitale de type 4Using a Type 4 Digital statement, , nous pouvons produire des caractères répétés ou une suite de caractères sur la molette RNG:

```
RNG 4 1000 a ^ b
```

Maintenant la seule manières d'arrêter cette séquence de cractères est le bouger la molette RNG dans sa position centrale, ou le caractère null (^) est généré. Mais disons qu'au lieu de cela, nous préférions relacher la molette RNG, et en exemple, utiliser le bouton T6 pour stopper ou démarrer la séquence de caractère. Avec la programmation logique, c'est très facile d'y parvenir avec ces instructions:

```
DEF X3 T6*
DEF X4 X1 AND NOT X3
DEF X5 X2 AND NOT X3
RNG 4 1000 X1 ^ X2
BTN X4 a
BTN X5 b
```

La façon dont cela fonctionne est la suivante. La molette RNG au lieu de produire une séquence de caractères fait passer à ON ou OFF les flags logiques X1 et X2. Les boutons X4 et X5 génèrent les caractères a et b, mais seulement si le flag logique X3 n'est pas ON, et comme le bouton T6 fait basculer X3 entre ON et OFF, nous avons maintenant une manière de mettre ON ou OFF les caractères générés par la molette RNG.

8.5.2 Une fonction de trim 'lent'

Merci à Mark ! que je remercie pour sa permission à présenter cette technique!
Nous considérons que dans un simulateur de vol, nous utilisons KP7 et KP1 pour trimmer vers le haut ou vers le bas l'avion. ce que nous voulons faire, est une fonction de trim 'lent', et donc si on presse le HAT1 vers le haut et que nous le relachons, alors toutes les 5 secondes, KP7 est généré pour ajuster le trim. Et de même pour le HAT1 vers le bas avec KP1. Voilà la solution :

```
DEF X1 H1U* AND (NOT X2)
DEF X2 H1D* AND (NOT X1)
```

```
BTN X1 /A KP7 DLY(5000)
BTN X2 /A KP1 DLY(5000)
```

Voyons ce qui se passe ici. Quand le Hat 1 est pressé vers le haut, il passe X1 à ON qui reste ON car nous avons utilisé le flag (*). Si X1 est ON alors nous exécutons l'instruction du bouton X1, et nous obtenons une répétition du caractère KP7 toutes les 5 secondes. Quand le Hat 1 est pressé de nouveau, il passe X1 à off et nous stoppons de générer KP7. C'est pareil pour X2. l'inclusion de (NOT X1) assure, que si le Hat 1 est pressé vers le bas et qu'il génére KP1 et si vous appuyez alors vers le haut, qu'il ne génére pas KP1 et KP7 au même moment.

Il y a des fichiers écrits par Mark Mooney dans votre répertoire avec d'autres exemples de programmation logique.

9. dépannages

9.1 Reset des controlleurs

Il ya plusieurs manières de traiter des problèmes avec les controlleurs, décrits ci-dessous.

9.1.1 Encours de jeu: EMPTY_BUFFERS et STICK_OFF

il est (c'est peu propable!) possible que vous généreriez trop de caractères depuis le Cougar, soit en pressant trop de boutons trop rapidement (peu probable), soit en utilisant un fichier mal programmé (ca c'est possible), entrainant une fonctionnement érratique du Joystick. La facon dont cela se manifeste est un arrêt de fonctionnement de tous les boutons bien que tous les axes continuent à fonctionner. Dans cette situation, il est possible d'effacer le buffer memoire, sans pour autant effacer le programme à l'intérieur du controlleur, vous pourrez donc continuer à jouer..

Syntaxe de la commande:

```
EMPTY_BUFFERS
```

Exemple:

```
BTN S2 EMPTY_BUFFERS
```

Objectivement, c'est quelquechose qui sera rarement utilisé, et il serait alors justifié de l'employer aucours d'une programmation logique demandant une séquence spécifique de caractères pressés en même temps, ce qui évitera une utilisation accidentelle.

```
DEF X1  DELAY(2000) S1 AND S4
BTN X1 EMPTY_BUFFERS
```

Dans l'exemple ci-dessus, vous devez maintenir enfoncé S1 et S4 ensemble, pendant 2 secondes, ce qui enverra une commande EMPTY_BUFFERS au controlleur.

D'une façon similaire, vous pouvez arrêter le controlleur encours de jeu, juste en activant le mode Windows mode pour les boutons et en utilisant l'instruction STICK_OFF

Syntaxe de la commande:

```
STICK_OFF
```

Exemple:

```
BTN S2 STICK_OFF
```

Objectivement, c'est quelquechose qui sera rarement utilisé, et il serait alors justifié de l'employer aucours d'une programmation logique demandant une séquence spécifique de caractères pressés en même temps, ce qui évitera une utilisation accidentelle..

```
DEF X1  DELAY(2000) S1 AND S4
BTN X1 STICK_OFF
```

Une fois que vous utilisez cette commande, il n'y a aucune manière de re-démarrer les sticks, et donc c'est seulement a être utilisé en cas d'urgence si pour certaines obscures raisons, votre controlleur commence à générer des caractères et que vous ne pouvez pas le stopper. Vous devrez donc quitter votre jeu et rechercher les causes de vos problèmes sous Windows.

9.1.2 Sous Windows

Si vous regardez le menu Cougar de Foxy vous verrez des items vous permettant de faire un reset de plusieurs aspects de votre controlleur. Nous allons lister ici dans l'ordre d'importance les diffèrentes étapes vous permettant de retrouver la stabilité des controlleurs!

1. Empty buffers (Effacer les buffers)

Exaxtement de la même manière qu'auparavant, vous pouvez effacer les buffers mémoire tout en gardant le programme en mémoire et en laissant les controlleurs dans leur mode actuel.

2. Disabling programmed mode (Désactiver les modes de programmation)

C'est très facile à l'intérieur de Foxy d'activer le mode Windows des controlleurs afin de stopper alors la génération de caractères, si pour une raison donnée, un déluge de caractères est généré. Si vous êtes malchanceux et que les caractères générés sont la fonction "F1" , alors vous apprécierez le nombre impressionant de fenêtre d'aide qui s'ouvriront le temps que vous atteigniez le port USB afin de débrancher et re-brancher les controlleurs!

3. Clear memory (Effacer la mamoire)

Le mode Window est activé et tous les programmes en mémoire sont effacés.

4. Flasher la mémoire

Dans le cas ou il y aurait un problème sérieux avec les controlleurs, qui seraient reconnus par Windows, mais dont aucun axe ou bouton ne fonctionnerait, ou si une nouvelle version du BIOS est disponible, vous pouvez flasher ou re-flasher la mémoire du BIOS.

5. Appeler le support technique!

Si les controlleurs ne sont pas reonnus du tout, soit les drivers natifs Windows ne sont pas installés correctement, soit il y a un sérieux problème matériel. Vous devez alors contacter le support technique pour qu'ils determinent si vos controlleurs doivent nous être retournés.

10. Annexes

Annexe 1.
Résumé des instructions Thrusmaster

Instructions Boutons et attributs instructions

Instruction	Acronyme	Description
BTN	Bouton	Défini le bouton à programmer. Qui peut être Hat 1 à 4, S1 à S4, T1 à T10, TG1 et 2.
REM	Remarque	Tout texte après une instruction REM est ignoré par le compilateur (commentaires, titres ,etc ,etc.)
RESET_TOGGLES	Reset bascule	Reset un bouton bascule au premier /T
REVERSE_TOGGLES	Inverse bascule	Produis une bascule en sens inverse
DLY	Delai	Ajoute un délai entre les caractères.
RPT	Repéte	Répéte un caractère ou une macro
( )	Groupe parenthèses	Groupe des caractères/macros ensemble pour diverses instructions incluant les instructions digitales
{ }	Groupe crochets	Groupe des caractères ensemble et les génére avant que leur code 'break' soit généré. (Similaire à rester enfoncé sur un groupe de touches.)
< >	Intervalle Parenthèses	Force l'instruction contenu entre à être exécutée entierement avant qu'une autre instruction soit exécutée
DX	Bouton DirectX	Boutons DirectX qui peuvent être programmés par une instruction bouton
KD KU	Touche appuyée Touche relachée	Permet un contrôle plus important des évenements 'touches appuyés', 'touches relachées'.
USB	Code clavier USB	Utilisé pour générer le code 'make' et 'brake' d'un caractère.
REVERSE_UD REVERSE_LR	Inverse direction controlleur	Inverse les directions d"un controlleur (comme pour des

REVERSE_DIR	instructions HAT)

Instruction	Acronyme	Description
NOHOLD, KP5	Utilise l'instruction USE HAT AS	NOHOLD stoppe les instructions HAT produisant des caractères maintenus. KP5 ajoute la position centrale KP5 à l'instruction USE HAT AS KEYPAD.
FORCED_CORNERS	Coins Hat	Oblige les positions de coin des HAT à fonctionner comme un Hat à 4 positions
S3_LOCK, S3_UNLOCK	Mode verrou S3, Mode normal S3	Mode verrou pour S3 Mode Normal S3
SHIFTBTN	Assigne S3	Assigne un bouton diffèrent à S3
POV*n*, (n = D, L,R,UL, DL,UR, DR)	Vision position hat positions	Permet un contrôle programmé de la vision des positions du hat
MOUSE_LB, MOUSE_RB, MOUSE_MB	Boutons souris	Permet un contrôle programmé des boutons souris

Attributs Slash et attributs instructions

Attribut	Acronyme	Description
/U, /M, /D	haut, milieu bas	Triple les positions programmables pour les hat/boutons (à part T7 et T8) en utilisant les position du switch dogfight de la manette de gaz.
/I, /O	Appuyé, relaché	Double les positions programmables pour les hat/boutons quand le bouton S3 du joystick est préssé (ne peut-être utilsé sur S3 lui-même)
/P, /R	Appuyé, relaché	Permet de programmer des hat/boutons selon que le bouton est soit pressé soit relaché.
/T	Bascule	Bascule entre diffèrents caractères/macros à chaque appui de bouton
/H	Maintien	Produit un ractère maintenu enfoncé tant que le bouton est maintenu. Peut-être combiné avec d'autres attributs.
/A	Auto-Repétition	Répète tout ce qui se trouve sur la ligne.

Instructions de configuration

Syntaxe	Description
USE MDEF	Identifie quel fichier macro contient les définitions de macro pour le fichier joystick courant.
USE *Btn* AS DXn	Assigne les boutons DirectX. Remplace la syntaxe PORTB1 IS
USE ALL_DIRECTX_BUTTONS	Assigne le Hat1 comme POV et tous les autres boutons comme des boutons DirectX
USE HAT*n* FORCED_CORNERS	Converti un hat 8 positions en un hat 4 positions, ainsi les positions de coin éxécute les instructions des positions adjacentes
USE HAT AS MOUSE, POV, ARROWKEYS, KEYPAD	Défini comment utiliser le HAT si il n'est pas programmé par des instructions BTN
USE RATE	Fréquence par défaut à laquelle les caractères sont généres/répétés.
USE S3_LOCK USE S3_UNLOCK	S3 se conduit comme un switch 'verrou'
USE S3 AS SHIFTBTN	Défini un bouton diffèrent à utilser pour S3 /I, /O
USE HATx_SENSITIVITY	Réduit la sensibilité aux positions de coin
USE T1_SENSITIVITY	Règle la sensibilité de T1
USE FOXY	Utilisé en interne par Foxy pour diverses fonctions. USE FOXY GRAPHIC USE FOXY README
USE NULLCHR	Défini quel sera le caractère NULL, par défaut '^'
USE KEYBOARD	AZERTY – utilise un clavier AZERTY en cas de problème d'assignement des touches
USE PROFILE	Utilise des profils sauvegardés depuis le panneau de contrôle du TM Cougar TM
USE CURVE	Défini une courbe de réponse d'un axe
DISABLE AXIS	Désactive un axe
USE SWAP	Echanges les axes
USE REVERSE	Inverse un axe
USE AXES_CONFIG	Défini quels axes à utiliser et leurs attributs
USE MTYPE	Assigne le contrôle de la souris au microstick et défini quels boutons à utiliser pour les boutons souris
USE MICROSTICK AS MOUSE - NO_BUTTON	Assigne le contrôle de la souris au microstick, aet le bouton gauche de la souris à T1 (à moins que NO_BUTTON soit présent)
USE Axis_Identifier AS Mouse_Axis	Assigne un axe pour contrôller la souris
USE ZERO_MOUSE	Evite des blocages souris avec /I, /O
DISABLE MOUSE	Désactive l'assignement par défaut de la souris
USE SCREEN_RESOLUTION	Utilisé pour des mouvements complexes de la souris

Programmation des axes

Instruction Axe	Acronyme	Description
JOYX, JOYY	Joystick X et Y	Axes Joystick
THR	Manette des gaz	Axe manette des gaz
RDDR	Pâlonnier	Axe palonnier
ANT	Molette antenne	Axe molette antenne
RNG	Molette RNG	Axe molette RNG
MIX, MIY	Microstick X et Y	Axe Microstick
LBRK and RBRK	Freins pied gauche et droit	Axes freins pieds
MSX, MSY, MSZ	Souris X, Y, Z	Axes souris - Z est la roulette souris
Instructions digitales de Type 1 à 6		Génére des caractères clavier depuis les axes. Utilisé aussi avec des instructions souris, des courbes et des instructions de flags logiques.
FORCE_MACROS	Force macros	Oblige les instructions caractères/macros de type 1, 2, 5, 6 à être toujours
CURVE	Courbe Axe	Courbes axes – Change la réponse de l'axe et la sensibilité
TRIM TO_CURRENT HOLDTRIM	Trim Axe	Trim un axe à une valeur spécifiée
LOCK, UNLOCK, LASTVALUE	Verrouille Axe	Maintien une valeur d'axe afin qu'il ne soit utilisé qu'en digital
SWAP	Echange Axes	Echange les axe.
REVERSE FORWARD	Direction Axe	Inverse un axe, retourne la direction normale

Instructions avancées Souris

Instruction Souris	Description
MOUSEXY	Positionne le curseur de la souris à une position écran determinée
MOUSEMOVE	Bouge la souris relativement à sa position actuelle
MOUSEROTATE	Programme la souris pour qu'elle bouge en mouvement circulaire, permettant le contrôle des cadrans circulaires d'un cockpit par programmation

Instructions logiques

Syntaxe logique	Acronyme	Description
DEF	Défini	Défini des flags logiques à travers des équations
BTN	Bouton	Utilisé pour assigner une instruction bouton virtuel à un bouton logique X1 à X32
X1 to X32	Flags logigues	Flags logiques qui sont soit ON soit OFF.
AND, OR, NOT	Comparateurs logiques	Utilisé pour construire des équations logiques
*	Bascule	Bascule un flag logique entre ses états ON et OFF
DELAY	Delai	Exécute un delai après qu'une condition logique soit Vraie
PULSE	Fréquence	Génére des états caractères/flags toutes les *nn* millisecondes

Instructions Matérielles

Syntaxe	Description
EMPTY_BUFFERS	Vide le buffer mémoire des controlleurs
STICK_OFF	Arrête le fonctionnement des controlleurs en cours de jeu

Appendice 2. Syntaxe touche Thrustmaster

ESC	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	
`	1	2	3	4	5	6	7	8	9	0	-	=	BSP
TAB	q	w	e	r	t	y	u	i	o	p	[	]	\
CAPS	a	s	d	f	g	h	j	k	l	;	'	ENT	
LSHF	z	x	c	v	b	n	m	,	.	/	RSHF		
LCTL	LALT	SPC	RALT	RCTL									

PRNTSCRN	SCRLCK	BRK

INS	HOME	PGUP
DEL	END	PGDN

	UARROW	
LARROW	DARROW	RARROW

NUML	KP/	KP*	KP-
KP7	KP8	KP9	KP+
KP4	KP5	KP6	
KP1	KP2	KP3	KPENT
KP0		KP.	

NOTES

1. *la syntaxe pour les touches combinées (quand vous maintenez enfoncé les touches Shift, ALT ou CTRL) est* SHF a, ALT b, CTL c. *Ce n'est pas* LSHF a, LALT b LCTL c

2. *Certaines touches sont réservées:* ( ) { } < > *et doivent être programmées avec l'instruction SHF:*

(= SHF 9
) = SHF 0
{ = SHF [
} = SHF]
< = SHF ,
> = SHF .

Annexe 3. Code USB majuscule et minuscule

Exemple:

BTN S2 /P USB (D04) Rem “a” Touche enfoncée
/R USB (U04) Rem “a” Touche relachée

Touche	Syntaxe TM	Code USB HID
a A	a	04
b B	b	05
c C	c	06
d D	d	07
e E	e	08
f F	f	09
g G	g	0A
h H	h	0B
i I	I	0C
j J	j	0D
k K	k	0E
l L	l	0F
m M	m	10
n N	n	11
o O	o	12
p P	p	13
q Q	q	14
r R	r	15
s S	s	16
t T	t	17
u U	u	18
v V	v	19
w W	w	1A
x X	x	1B
y Y	y	1C
z Z	z	1D
1 !	1	1E
2 @	2	1F
3 #	3	20
4 $	4	21
5 %	5	22
6 ^	6	23
7 &	7	24
8 *	8	25
9 (	9	26
0)	0	27
Return	ENT	28
Escape	ESC	29

Backspace	BSP	2A
Tab	TAB	2B
Space	SPC	2C
- _	-	2D
= +	=	2E
[{	[	2F
] }	]	30
\ |	\	31
Europe 1 (*voir notes*)		32
; :	;	33
' "	'	34
` ~	`	35
, <	,	36
. >	.	37
/ ?	/	38
Caps Lock	CAPS	39
F1	F1	3A
F2	F2	3B
F3	F3	3C
F4	F4	3D
F5	F5	3E
F6	F6	3F
F7	F7	40
F8	F8	41
F9	F9	42
F10	F10	43
F11	F11	44
F12	F12	45
Print Screen	PRNTSCRN	46
Scroll Lock	SCRLCK	47
Break (Ctrl-Pause)	BRK	48
Pause	BRK	48
Insert	INS	49
Home	HOME	4A
Page Up	PGUP	4B
Delete	DEL	4C
End	END	4D
Page Down	PGDN	4E
Right Arrow	RARROW	4F
Left Arrow	LARROW	50
Down Arrow	DARROW	51
Up Arrow	UARROW	52
Num Lock	NUML	53
Keypad /	KP/	54
Keypad *	KP*	55
Keypad -	KP-	56
Keypad +	KP+	57

Keypad Enter	KPENT	58
Keypad 1 End	KP1	59
Keypad 2 Down	KP2	5A
Keypad 3 PageDn	KP3	5B
Keypad 4 Left	KP4	5C
Keypad 5	KP5	5D
Keypad 6 Right	KP6	5E
Keypad 7 Home	KP7	5F
Keypad 8 Up	KP8	60
Keypad 9 PageUp	KP9	61
Keypad 0 Insert	KP0	62
Keypad . Delete	KP.	63
Europe 2 (*Voir notes*)		64
Keypad =		67
F13		68
F14		69
F15		6A
F16		6B
F17		6C
F18		6D
F19		6E
F20		6F
F21		70
F22		71
F23		72
F24		73
Keyboard Execute		74
Keyboard Help		75
Keyboard Menu		76
Keyboard Select		77
Keyboard Stop		78
Keyboard Again		79
Keyboard Undo		7A
Keyboard Cut		7B
Keyboard Copy		7C
Keyboard Paste		7D
Keyboard Find		7E
Keyboard Mute		7F
Keyboard Volume Up		80
Keyboard Volume Dn		81
Keyboard Locking Caps Lock		82
Keyboard Locking Num Lock		83
Keyboard Locking Scroll Lock		84
Keypad , (Brazilian Keypad .)		85
Keyboard Equal Sign		86
Keyboard Int'l 1 (Ro)		87

Keyboard Intl'2 (Katakana/Hiragana)		88
Keyboard Int'l 2 ¥ (Yen)		89
Keyboard Int'l 4 (Henkan)		8A
Keyboard Int'l 5 (Muhenkan)		8B
Keyboard Int'l 6 (PC9800 Keypad ,)		8C
Keyboard Int'l 7		8D
Keyboard Int'l 8		8E
Keyboard Int'l 9		8F
Keyboard Lang 1 (Hanguel/English)		90
Keyboard Lang 2 (Hanja)		91
Keyboard Lang 3 (Katakana)		92
Keyboard Lang 4 (Hiragana)		93
Keyboard Lang 5 (Zenkaku/Hankaku)		94
Keyboard Lang 6		95
Keyboard Lang 7		96
Keyboard Lang 8		97
Keyboard Lang 9		98
Keyboard Alternate Erase		99
Keyboard SysReq/Attention		9A
Keyboard Cancel		9B
Keyboard Clear		9C
Keyboard Prior		9D
Keyboard Return		9E
Keyboard Separator		9F
Keyboard Out		A0
Keyboard Oper		A1
Keyboard Clear/Again		A2
Keyboard CrSel/Props		A3
Keyboard ExSel		A4
Left Control	LCTL	E0
Left Shift	LSHF	E1
Left Alt	LALT	E2
Left GUI		E3
Right Control	RCTL	E4
Right Shift	RSHF	E5
Right Alt	RALT	E6
Right GUI		E7

NOTES

Ces touches peuvent avoir des comportements diffèrents selon la localisation du clavier; Europe 1 est un clavier AT-101 touches 42 à coté de la touche ENTREE . Europe 2 est un clavier AT-101 touches 45, entre le shift gauche et la touche Z.

Appendix 4. Differences entre les anciens fichiers TM et les fichiers Cougar

Il y a des diffèrences subtiles (et d'autres non) avec les anciens fichiers TM supportant les puces digitales, F22, FLCS, FCS, TQS, WCS MkII. cette appendice résume ces diffèrences – Voir la documentation trhusmaster pour des explications plus détaillées.

1. Changements dans la syntaxe des touches

Ancienne syntaxe	Nouvelle syntaxe
LSFT	LSHF
RSFT	RSHF
none	PRNTSCRN
AUXUAROW, UAROW	UARROW
AUXDAROW, DAROW	DARROW
AUXLAROW, LAROW	LARROW
AUXRAROW, RAROW	RARROW
AUXENT	KPENT
AUX/	KP/
AUXINS	INS
AUXHOME	HOME
AUXPGUP	PGUP
AUXPGDN	PGDN
AUXDEL	DEL
AUXEND	END

2. Changements des attributs '/'

Attribut '/'	Commentaires
/U	Comme avant
/M	
/D	
/I	Comme avant mais l'instruction /I soit toujours apparaître avant l'instruction /O, et /I, /O doivent être sur des lignes diffèrentes
/O	
/P	Comme avant
/R	
/T	Comme avant
/A	Défini désormais des instructions auto-répétés
/H	Comme avant mais les caractères sont générés de facon répété, et dans des instructions complexes, s'applique à la dernière instruction
/F	Désormais non supporté
/Q	Désormais non supporté
/N	Désormais non supporté – toutes les instructions se comportent comme si elles n'etaient jamais répétées tant que les attributs /H, /A ne sont pas utilisés

3. Instructions désormais non supportées

Instruction	Commentaires
RAW ( )	Remplacé par USB ( ) *(HID codes)* mais plus facile à utiliser les instructions KD( ) et KU( )
BTN MT	Utilisez une instruction Type 5 plus puissante
BTN T11 – T14 (they no longer exist)	Le microstick est comme un joystick 2 axes, ce n'est pas un controlleur 4 boutons. Donc il peut-être programmé digitalement avec des instructions digitales Type 1 à 6
USE RCS	Plus besoin désormais
USE TQS	Plus besoin désormais
USE WCS	Plus besoin désormais
USE RCSPRO	Plus besoin désormais
USE NOMOUSE	Plus besoin désormais
USE NOTHR	Plus besoin désormais
USE MTYPE (B or C)	Types de souris désormais non supportés, des instructions 3 boutons peuvent être générées à la place.

4. Extension de fichier, noms de fichier

Le fichier joystick doit finir par ".tmj" et le fichier macro par ".tmm". De plus, les fichiers joystick et macros peuvent contenir des caracères espace et ne sont plus limités à 8 caractères. Mais comme auparavant, le fichier joystick doit-être dans le même repertoire que le fichier macro. Par défaut c'est le repertoire de Foxy.

5. Actions par défaut

le compilateur va utiliser quelques actions par défaut dans vos fichiers, dependant de vos préférences renseignées dans Foxy. Ces options sont contredites soit si ils sont deselectionnées sous Foxy, soit si vous avez progrémme vos fichiers pour réagir diffèremment.
Ceux sont:

- Hat 1 ou un Hat de votre choix comme hat de point de vision
- TG1 comme DX1, S2 comme DX2 – comme boutons DirectX, donc ces boutons auront leurs fonctions assignés assignés dans le jeu si le jeu les supporte.
- Si aucune ligne USE MDEF n'est présente et si le fichier joystick contient des macros, alors le compilateur cherchera un autre fichier ayant le même nom que le fichier joystick , mais avec l'extension ".tmm"
- Le microstick controllera les mouvements souris, avec le bouton T1 du microstick fonctionnant comme le bouton gauche de la souris.

Ces pramaètres par défauts facilitent la vie du débutant lorsqu'il début en programmation. Ainsi un utilisateur qui a un fichier joystick doit juste écrire :

BTN S1 Autopilot

Et un fichier macro avec juste :

Autopilot = a

Et il sera capable de charger le fichier joystick et tout le reste fonctionnera, la souris fonctionnera …etc..etc..
Le seul danger ici est que si le fichier est donné à quelqu'un d'autre qui a changé ses paramètres par défaut dans Foxy, stipulant qu'il faut produire une erreur si une ligne USE MDEF n'est pas trouvée, alors ce fichier ne fonctionnera pas pour cet utilisateur..

6. Axes digitales vc axes analogiques.

Avec les anciens fichiers TM, un axe pouvait être programmé soit digital, produisant des caractères claviers, soit analogique (mode par défaut), par lequel moyen cette fonction etait assignée dans le jeu. (manette des gaz = poussée des réacteur dans unsimulateur de vol). Avec le HOTAS cougar, les axes peuvent être à la fois digitale et analogiques au même moment.. Pour avoir un axe purement digital, ces fonctions analogiques doivent être désactivées avec l'instruction DISABLE. Notons aussi qu' avec le Cougar, vous n'avez rien besoin de changer dans le panneau de configuration des options du jeu, si vous voulez

utiliser un axe puremment digital, à l'inverse de ce que vous deviez faire avec les anciens controlleurs TM.

7. Instructions digitales Type 1

La syntaxe a changée, voir la section detaillant cela dans le présent manuel de référence.

8. Manette des gaz non présente

Si vous avez un fichier écrit pour un joystick et une manette de gaz, et que la manette des gaz n'est pas présente, alors avec les attributs instructions /U, /M, /D , seule l'attribut /M sera compilé
/U, /D seront ignorés. Les instructions axes de la manette de gaz seront aussi ignorées.

9. Macros – caractères interdits

Vous ne **pouvez pas** utiliser les caractères suivants dans les noms de macro:

= < > { } ( ) ^ , *spaces*

10. RPT

si vous avez une macro comme: Macro1 = a b c

et une instruction comme cela: BTN S2 RPT (3) Macro1

alors quand S2 sera préssé, vous aurez: a a a b c

Pour éviter cela, entourez la macro ou les caractères avec des parenthèses:

BTN S2 RPT (3) (Macro1)

ou

BTN S2 RPT (3) Macro1

avec: Macro1 = (a b c)

11. le caractère de commentaire //

Avec les puces digitales, vous pouvez utiliser le caractère // au lieu de l'instruction REM. Ce n'est pas supporté par Foxy.